Proposition d'une modélisation unifiée du Cloud Manufacturing et d'une méthodologie d'implémentation, basées sur les ontologies d'inférence

Asma Talhi

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech

2016-ENAM-0017

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " génie industriel "

présentée et soutenue publiquement par

Asma TALHI

31mai 2016

Proposition d'une modélisation unifiée du Cloud Manufacturing et d'une méthodologie d'implémentation, basées sur les ontologies d'inférence

Directeur de thèse : **Samir LAMOURI**
Co-encadrement de la thèse : **Jean-Charles HUET**

Jury

M. Gilles Gonclaves Professeur des Universités Université d'Artois	Président
M. Kary Främling Professeur - Aalto University	Rapporteur
M. Dimitris Kiritsis Professeur - EPFL	Rapporteur
M. Pierre Féniès Professeur des Universités Université Paris Ouest	Examinateur
Mme Virginie Fortineau Présidente-directrice générale Khtema SAS Jeune Entreprise Innovante	Examinateur
M. Jean-Charles Huet Maître de Conférences - ECAM-EPMI	Examinateur
M. Samir Lamouri Professeur des Universités Arts et Métiers Paristech	Examinateur

T H È S E

Arts et Métiers ParisTech - Campus de Paris
LCPI

À mes proches…

« Le savoir scientifique n'est pas absolu, mais socialement, culturellement, technologiquement et historiquement marqué, donc provisoire. »

Steven Rose

Remerciements

En premier lieu, je remercie Gilles Goncalves de me faire l'honneur de présider le jury scientifique. Je remercie Kary Främling et Dimitris Kiritsis d'avoir accepté de relire cette thèse et d'en être rapporteurs. Merci à Pierre Féniès dont les commentaires ont permis d'améliorer mon rapport.

A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate.

Je tiens à exprimer mes plus vifs remerciements à Samir Lamouri qui fut pour moi un directeur de thèse attentif et disponible malgré ses nombreuses charges. Ses qualités pédagogiques et scientifiques, sa clairvoyance et sa franchise ont été des moteurs de mon travail de recherche.

J'adresse de chaleureux remerciements à mes encadrants de thèse, Jean-Charles Huet et Virginie Fortineau, qui ont toujours été à l'écoute de mes nombreuses questions, et se sont toujours intéressés à l'avancée de mes travaux. Jean-Charles, pour sa disponibilité sans faille, ses remarques pertinentes et son aide précieuse et constante. Virginie Fortineau, sa rigueur, sa capacité à analyser les problèmes et ses nombreuses connaissances m'ont permis de progresser et ont répondu à plusieurs de mes préoccupations. Merci d'avoir rendu la continuité de cette aventure possible en m'accueillant au sein de l'entreprise KHTEMA.

Je tiens à remercier M. Xavier Lopez, PDG de l'entreprise WYSIWYG et initiateur de ce contrat CIFRE, sans qui cette thèse n'aurait pas vu le jour. Je remercie M. Alain Legrand de la société ALFAP pour la qualité scientifique et intellectuelle des échanges qu'il nous a permis d'avoir.

Mes pensées et remerciements vont aux chercheurs et enseignants de l'équipe « Logil » et aussi des arts et métiers pour ces discussions qui m'ont aidée dans ma réflexion : Albéric, Alain, Simon et Jean-Baptiste.

Je voudrais adresser un grand merci à mes amis qui m'ont supportée durant cette thèse. Un merci particulier à Laure pour sa présence et à Franck pour son soutien indéfectible. Enfin, mes profonds remerciements vont à ma famille. Mes parents, pour les valeurs qu'ils m'ont inculquée, leur patience et leur amour sont pour beaucoup dans l'achèvement de ce travail. Mes sœurs ainsi que le reste de la famille pour leurs encouragements. Cette thèse vous est dédiée.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

ABC	Activity Based Coasting
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
ASCI	Analyse, Spécification, Conception, Implémentation
ASP	Application Service Provider
BOL	Beginning Of Life
BPM	Business Process Management
CC	Cloud Computing
CM	Cloud Manufacturing
CMO	Cloud Manufacturing Ontology
CRM	Customer Relationship Management
EOL	End Of Life
ERP	Enterprise Resource Planning
GC	Grid Computing
GSL	Global Service Layer
Haas	Hardware as a Service
HMS	Holon Manufacturing System
Iaas	Infrastructure as a Service
IETF	Internet Engineering Task Force
MBCSA	Méthodologie Bidirectionnelle de Conception des Systèmes Automatisés
MDA	Model Driven Architecture
MGrid	Manufacturing Grid
MOL	Middle Of Life
MO-MRSCOS	Multi-Objectives MGrid Resource Service Composition and Optimal-Selection
MRL	Manufacturing Resource Layer
NIST	National Institute of Standards and Technology
OPaas	Open Platform as a Service
OWL	Ontologie Web Language
Paas	Platform as a Service
PC	Personal Computer
PDM	Product Data Management

PERA	Purdue Enterprise Reference Architecture
PLM	Product Lifecycle Management
PME	Petites et Moyennes Entreprises
QoS	Quality Of Service
REST	Representational State Transfer
RFID	Radio Frequency IDentification
RDF	Resource Description Framework
Saas	Software as a Service
SAM	Software Asset Management
SI	Système d'Information
SIE	Système d'Information d'Entreprise
SLA	Service-Level Agreement
SOA	Service Oriented Architecture
STEP	STandard for the Exchange of Product model data
SysML	Systems Modeling Language
TIC	Technologies de l'Information et de la communication
VM	Virtual machine
VPC	Virtual Private Cloud
VPN	Virtual Private Network
VSL	Virtual Service Layer
UD	User Domain
UML	Unified Modeling Language
URI	Unified Resource Identifier
UX	User eXperience
XML	Extensible Markup Language

Contexte de recherche et objectifs de la thèse

WYSIWYG est une PME de Cergy-pontoise proposant des services aux entreprises comme des outils de gestion de production, de gestion commerciale et de comptabilité, et propose aussi des services d'externalisation de systèmes d'information vers des solutions basées sur le concept de Cloud Computing. J'ai été recrutée en 2011 à WYSIWYG par Xavier LOPEZ, PDG, qui cherchait à lancer une thèse CIFRE pour pouvoir proposer de nouveaux services alliant ces domaines de compétences. Un travail de prospection a donc dû être nécessaire pour vérifier l'existant : c'est pourquoi un état de l'art poussé a été réalisé dans les premiers chapitres de cette thèse. C'est dans ce cadre que j'ai souhaité mener un travail de recherche, officialisé par une inscription en thèse auprès de l'École Doctorale « Sciences des Métiers de l'Ingénieur » (SMI) rattachée à l'École nationale supérieure d'arts et métiers (ENSAM) - Arts et Métiers ParisTech - au mois de mai 2012. Cette thèse se situe dans la lignée des travaux effectués par l'équipe du Pr Samir LAMOURI [Lamouri 06] à l'ENSAM autour du *Product Lifecycle Management* (PLM). Ces travaux sur cette thématique ont été initiés par [Paviot 11] et [Fortineau 13a]. Ce travail de recherche a été effectué en collaboration avec une école du même territoire que l'entreprise : ECAM-EPMI dont des travaux sur la modélisation des systèmes complexes ont été menés [Huet 11]. Le travail présenté dans ce mémoire se situe aux interfaces des thématiques « PLM » et « Cloud Computing », c'est-à-dire une vision du PLM où toutes les phases sont vues comme un ensemble de services mis à la disposition des utilisateurs sur internet.

L'environnement industriel est en concurrence croissante et l'un des défis auxquels les entreprises font face aujourd'hui est de satisfaire la demande des clients en quête de produits de plus en plus sophistiqués tout en réduisant le coût et le délai de développement. Afin d'atteindre cet objectif, les entreprises mettent l'accent aujourd'hui sur la collaboration en établissant des partenariats avec d'autres organisations pour bénéficier de leurs compétences et infrastructures. Ce modèle manufacturier permettant aux acteurs impliqués dans le développement du produit de collaborer tout au long de son cycle de vie est appelé l'entreprise en réseau.

Le PLM [Terzi 05] s'est imposé comme une approche permettant la gestion des données relatives au produit dans de tels environnements collaboratifs. Le succès d'une approche PLM exige que le système sous-jacent joue le rôle d'un médiateur afin d'assurer la communication et l'échange de données entre les partenaires du projet ; fournisseurs, clients, etc. Par conséquent, un accès universel à l'information du produit à travers toutes les étapes du cycle de vie est essentiel pour assurer une collaboration efficace entre les parties prenantes. D'autre part, avec la volatilité de la demande des consommateurs, il est nécessaire que le système mis en place suive les modifications et s'adapte aux changements de la demande et du marché. Nous avons ainsi identifié le besoin d'avoir une architecture distribuée qui servira de support pour

de tels systèmes. Cette architecture doit assurer une flexibilité, évolutivité et robustesse aux systèmes PLM, afin de répondre aux exigences des systèmes actuels.

Les technologies et outils web sont largement utilisés aujourd'hui, non seulement dans la vie de tous les jours, mais aussi dans le milieu industriel. Les services webs et le *Cloud Computing* (CC) sont des exemples de telles technologies. Afin de bénéficier des *Technologies de l'Information et de la communication* (TIC), des initiatives ont été menées et nous avons constaté dans les travaux l'émergence de solutions orientées web. Nous avons identifié le *Cloud Manufacturing* (CM) (Figure1) comme solution prometteuse car, comme le CC, il tente d'apporter une solution où toutes les ressources et capacités impliquées dans le cycle de vie sont fournies à l'utilisateur sous la forme d'un service. La condition de faisabilité de ce type de systèmes est que les usines et ateliers possèdent une partie informationnelle développée afin qu'ils puissent être intégrés avec les reste des acteurs du cycle de vie. Par conséquent ces ateliers et usines doivent faire partie d'un *Holon Manufacturing System* (HMS). En effet, un *holon* est une entité autonome possédant une partie physique et une partie logique communicant et coopérant avec son environnement. Dans l'exemple (figure 1) l'ensemble des services requis tout au long du cycle de vie sont offerts par différents fournisseurs. La mise en correspondance entre l'utilisateur et cet ensemble de services est assurée par une plate-forme CM via laquelle l'utilisateur émet une requête afin de pouvoir utiliser un ou plusieurs services selon ses besoins.

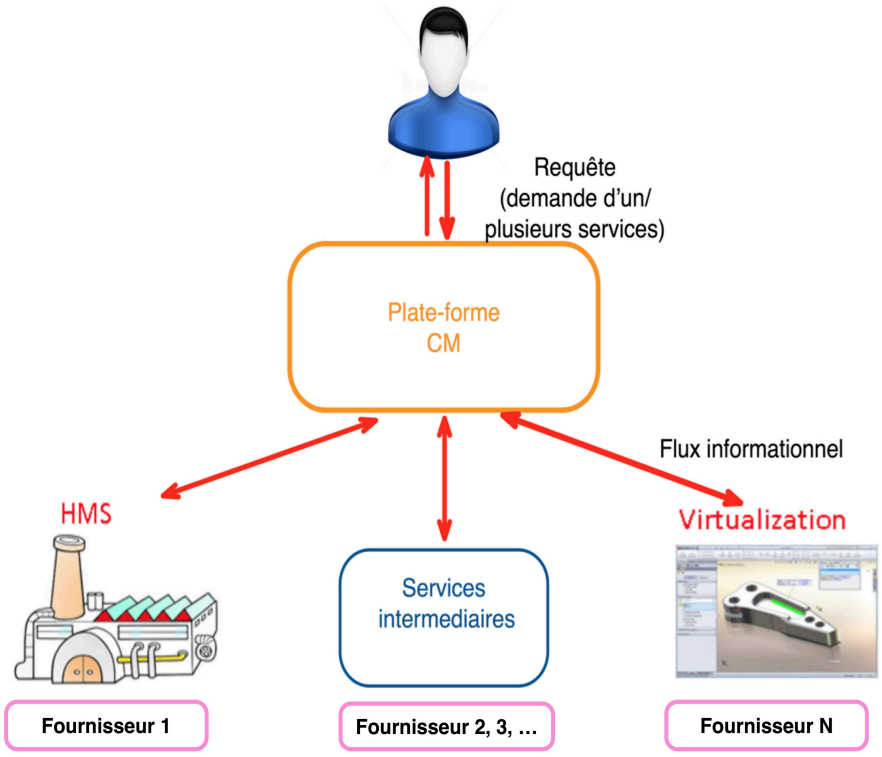


FIGURE 1 – *Illustration du fonctionnement d'une plate-forme CM*

Par conséquent, nous nous plaçons à l'intersection d'un nombre de disciplines : PLM, CC/CM et des systèmes manufacturiers intelligents afin de modifier la vision actuelle des *Système d'Information d'Entreprise* (SIE) et converger vers des modèles où le SIE est proposé

comme un service. Dans leurs travaux de thèses, [Paviot 11] et [Fortineau 13a] se sont focalisés sur l'interopérabilité sémantique des systèmes afin de permettre l'élaboration d'une démarche PLM. L'interopérabilité étant la capacité de deux systèmes ou plus à collaborer, et doit être réalisée d'un point de vue sémantique, technique et organisationnel [EIF 04]. Dans ces travaux, qui se veulent la continuité des travaux de doctorat de Thomas Paviot et Virginie Fortineau, nous traitons l'aspect technique du problème et postulons que le CM permet de mettre en place des systèmes flexibles, évolutifs et robustes. Pour mettre en place un tel système, nous avons identifié le besoin de suivre une méthodologie pour la modélisation d'un domaine et la réalisation de l'architecture sous-jacente. Cette méthodologie permettra non seulement d'expliquer les étapes nécessaires au développement d'un système CM, mais aussi servira de guide pour les utilisateurs souhaitant migrer leurs SIE vers le *cloud*, ou encore développer un système similaire en suivant les étapes décrites dans la méthodologie mais en choisissant d'autres méthodes ou outils.
Notre objectif est de proposer une architecture CM, en nous appuyant sur une méthodologie globale et réutilisable par l'ensemble des entreprises souhaitant migrer leurs SIE vers une solution *cloud*. La méthodologie choisie s'articule autour de deux étapes majeures ; la première permettant la proposition d'un modèle générique de connaissance permettant de représenter le domaine CM et la seconde, le développement d'une bibliothèque de composants informatiques basée sur le modèle générique proposé.
Afin de présenter la suite logique des travaux décrits tout au long de ce mémoire, un schéma est proposé (Figure 2)

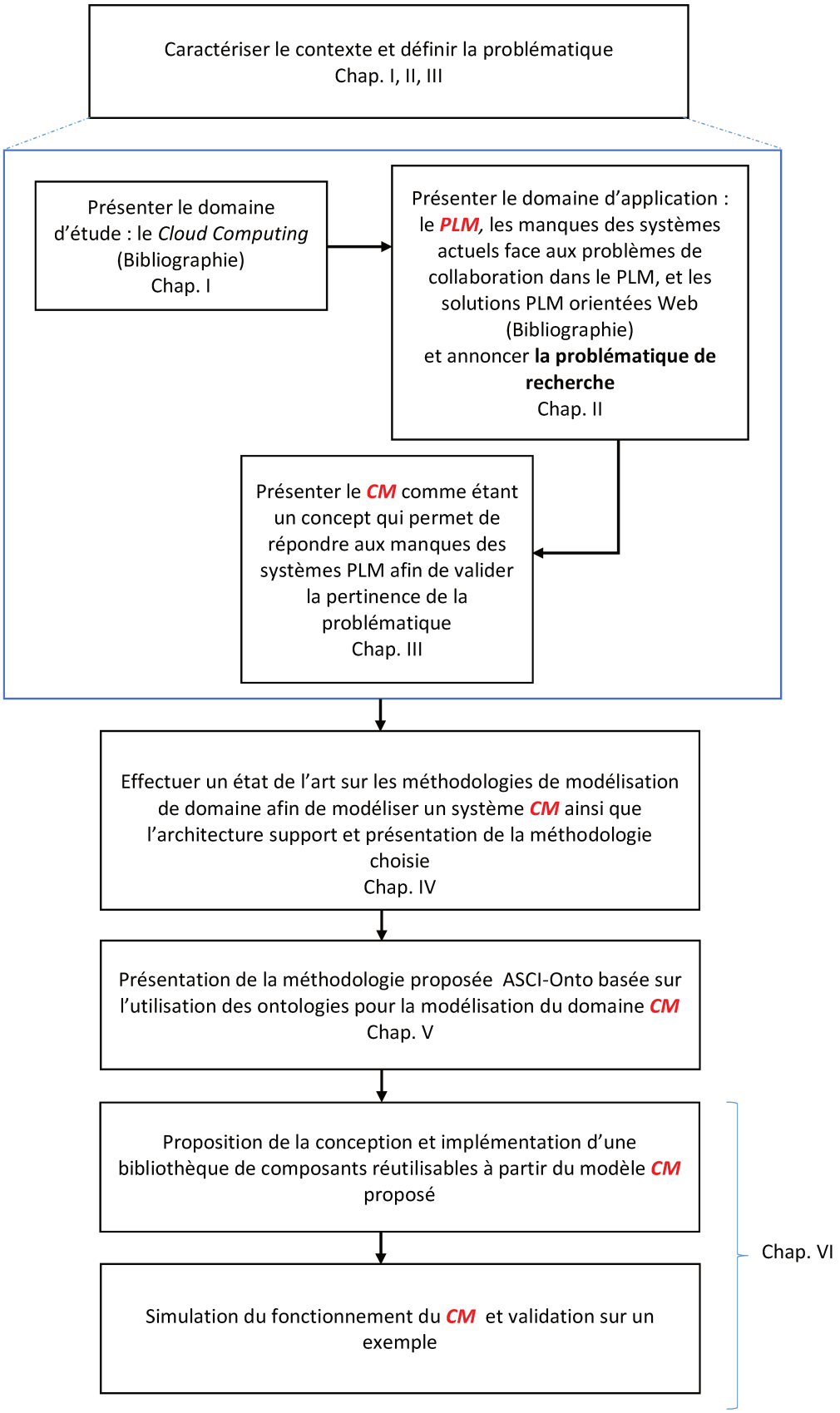


FIGURE 2 – *Articulation de la recherche*

Le ***premier chapitre*** présente, à travers une revue de la littérature, le concept du CC. Dans cette partie nous confrontons les points de vue des scientifiques et des industriels quant à la définition du CC. Ce chapitre introduit les modèles de services du CC : *Software as a Service* (Saas), *Platform as a Service* (Paas), *Infrastructure as a Service* (Iaas) ainsi que les caractéristiques et avantages du CC. Il décrit également les étapes de l'émergence de ce paradigme et les scénarii où il s'avère bénéfique d'adopter une solution *cloud* et pour quel type d'entreprise. Les technologies permettant la mise en place du CC et les exigences architecturales sont explorées afin de montrer les attentes des acteurs de ce type de systèmes. Ce premier chapitre donne une partie du positionnement général de notre recherche : il présente notre domaine d'étude.

Dans le ***deuxième chapitre*** nous présentons notre domaine d'application : le PLM. Ce chapitre permet de délimiter le terrain que nous envisageons d'étudier. Nous exposerons l'objectif du PLM ainsi que les modèles d'entreprises ayant permis le développement collaboratif des produits : les entreprises en réseau. Nous présentons ensuite les problèmes de collaboration dans de tels environnements. Ces problèmes ont été identifiés par la communauté scientifique comme étant le manque de flexibilité, d'évolutivité et de passage à l'échelle et, d'accès omniprésent aux ressources manufacturières. Cette analyse nous permet d'introduire le CM et d'aborder ensuite notre problématique de recherche : **définir une méthodologie qui permet de mettre en place une plate-forme Cloud Manufacturing et ainsi servir de support aux systèmes industriels traditionnels pour migrer vers le Cloud**.

Le ***chapitre 3*** présente en détails le *Cloud Manufacturing* (CM) qui est un concept où toutes les ressources manufacturières sont fournies à l'utilisateur sous la forme d'un service via Internet. Ces ressources sont d'origines matérielles ou logicielles et représentent l'ensemble d'outils, méthodes, expertises, etc requis tout au long du cycle de vie du produit. Ces ressources sont représentées à l'utilisateur sous forme de services : design as a service (Daas), manufacturing as a service (MFGaas), experimentation as a service (Eaas), simulation as a service (SIMaaS), management as a service (Maas), maintain as a service (MAaas) et integration as a service (INTaas). Ainsi, nous démontrons qu'un service de type CM peut englober les trois types de services dans le CC (Saas, Paas, Iaas). Par exemple, un service de type Daas peut être la combinaison d'un logiciel de conception (Saas) et d'un support de stockage de données (Iaas). Ce chapitre permet aussi d'introduire l'architecture d'un système CM telle qu'elle est décrite dans la littérature suivie par une revue des travaux scientifiques proposant des solutions orientées Web afin de vérifier s'il existe une implémentation complète du CM. Nous considérons une implémentation complète, une solution qui intègre toutes les couches d'une architecture CM. Cette revue de la littérature nous permettra d'introduire dans le chapitre suivant la méthodologie envisagée pour modéliser le domaine CM et mettre en place une telle architecture.

Le ***chapitre 4*** présente un état de l'art sur les méthodologies de modélisation. Lorsque nous avons posé notre problématique de recherche, nous avons identifié le besoin d'avoir une méthodologie qui nous permettra dans un premier lieu à modéliser le domaine CM et à implémenter l'architecture sous-jacente et qui servira en deuxième lieu de guide aux utilisateurs et entreprises qui veulent migrer vers le *cloud*. Notre choix s'est porté sur la démarche *Analyse, Spécification, Conception, Implémentation* (ASCI) qui permet de concevoir une méthodologie de modélisation d'un domaine et son environnement logiciel. La démarche ASCI se compose de quatre étapes : les étapes d'Analyse et de Spécification permettent d'obtenir le modèle géné-

rique de connaissance tandis que les phases de Conception et d'Implémentation permettent la création de la bibliothèque de composants logiciels réutilisables.

Le ***cinquième chapitre*** reprend la démarche choisie dans le chapitre 4 (ASCI) et expose en détail notre proposition méthodologique dérivée d'ASCI que nous avons appelé : **ASCI-Onto**. Nous justifions ensuite notre choix pour les outils de modélisation : les ontologies. Les ontologies sont un outil de modélisation issues du web sémantique qui permettent de structurer les données sous-jacentes au web afin que celles-ci soient compréhensibles et traitables par les machines. L'utilisation de cet outil fait la particularité de notre proposition car c'est la description (sens) du service qui est prise en compte et non pas le terme (nom) qui lui est donné par le fournisseur. Nous présentons ainsi les ontologies et les raisons qui nous ont orienté vers un tel outil pour produire notre modèle générique de connaissance. Cette méthodologie permet de modéliser le domaine CM et sera réutilisable pour la modélisation de n'importe quel système faisant partie de cette classe de systèmes.

Le ***chapitre 6*** traite de la validation de la formalisation proposée. Cette validation représente la deuxième étape majeure de notre méthodologie et consiste à développer une bibliothèque de composants informatiques réutilisables. Afin de réaliser la bibliothèque des composants, une dérivation de l'ontologie proposée vers le langage UML est nécessaire. Bien que cette dérivation fasse perdre à notre modèle sa richesse sémantique et sa capacité d'inférence, elle représente une étape primordiale pour l'implémentation de l'architecture CM car elle permet de représenter la structure statique d'un système et les composants à implémenter. Ce chapitre présente aussi un cas d'application montrant le fonctionnement de la plate-forme CM en utilisant les classes implémentées et un simulateur de machine CNC hébergé sur un serveur local.

La conclusion dresse un bilan de la contribution de cette thèse et présente plusieurs perspectives à ces travaux de recherche.

1

Le Cloud Computing

Résumé du chapitre

Ce chapitre introduit, à travers une revue de la littérature, le concept du Cloud Computing (CC), ses modèles et ses avantages, notamment pour les applications industrielles et ce, vis à vis des paradigmes antérieurs que sont les Clusters et le Grid computing. Ainsi, la section 1.1 présente une chronologie des contributions qui ont donné naissance au CC et à l'informatique en tant que service. Ensuite, en section 1.2 sont définis les Clusters et le Grid computing. En synthèse, nous proposons en conclusion de cette section une double définition du CC : selon les acteurs économiques et ensuite selon son contenu formel, à partir d'une confrontation des contributions de la littérature scientifique.

Par la suite, un état de l'art sur le CC est proposé, et structuré suivant trois thématiques complémentaires : les modèles de services (section 1.3), le déploiement (section 1.4), et les architectures supports du CC (section 1.5). Les aspects architecturaux sont abordés suivant les points de vue du fournisseur de Cloud, de l'entreprise adoptant le Cloud et de l'utilisateur final.

Enfin, les sections 1.6 et 1.7 discutent respectivement des avantages et enjeux du CC suivies par la section 1.8 qui conclut ce chapitre.

Sommaire

1.1 La genèse du Cloud Computing

Tout au long de ces 60 dernières années, les systèmes informatiques ont évolué de façon spirale vers l'intégration et la distribution des ressources. Cette évolution est marquée par une transition des ordinateurs centralisés appelés *mainframes* [1] dans les années 70 vers une décentralisation des systèmes et l'émergence des ordinateurs personnels. Cependant, durant les dix dernières années, ces systèmes ont convergé vers un modèle basé sur la consolidation et le partage des ressources appelé *Cloud Computing* (CC).

Leonard Kleinrock, l'un des principaux responsables scientifiques du projet *Advanced Research Projects Agency Network* (ARPANET) qui a enrichi Internet, a expliqué en 1969 que [Kleinrock 05] : *les réseaux informatiques n'en sont qu'à leur début, mais à mesure qu'ils grandissent et deviennent complexes, nous verrons probablement la propagation des « services informatiques » qui, comme les services d'électricité et de téléphonie, desserviront les maisons et bureaux individuels à travers le pays*. Par cette vision de l'informatique il a en effet prévu la transformation majeure des systèmes informatiques qui a eu lieu, où les produits informatiques sont devenus accessibles à la demande des utilisateurs, et proposés sous forme d'un service. Nous employons ici le terme « service » dans le sens de la définition proposée par Paviot [Paviot 10] :

Définition 1.1 *Un service, du point de vue des systèmes d'information, est l'action demandée à un système (fournisseur de services) ou réalisé par lui, afin de satisfaire une partie du besoin du système client [Paviot 10].*

Pour parachever cet « informatique à la demande », plusieurs paradigmes ont été proposés et adoptés au cours des dernières décennies. Voas et Zhang [Voas 09] expliquent que d'un point de vue historique, cette transformation de l'informatique a été réalisée en six étapes (Figure 1.1) :

- phase 1 (1960) : les utilisateurs se servaient de terminaux pour se connecter à des ordinateurs centraux et puissants. Les terminaux, à cette époque, consistaient en des moniteurs et claviers ;
- phase 2 (1970) : les ordinateurs personnels (*Personal Computer* (PC)) sont devenus assez puissants pour répondre aux besoins quotidiens des utilisateurs qui, par conséquent, ont abandonné l'usage des ordinateurs centraux ;
- phase 3 (1980) : les PC et ordinateurs portables ont été connectés via un réseau local pour partager les ressources (telles que les imprimantes) ;
- phase 4 (1990) : cette phase a vu l'avènement d'Internet où des réseaux locaux pouvaient être connectés à d'autres réseaux locaux pour former un réseau global afin d'utiliser des ressources et des applications distantes ;
- phase 5 (2000) : un nouveau concept s'inspirant des réseaux électriques a vu le jour : « la grille informatique » (*Grid Computing* (GC)) qui est une consolidation de plusieurs ressources telles que des calculateurs et super-ordinateurs pour offrir une puissance de calcul et de stockage importante ;
- phase 6 (2010) : le CC fournit en outre un accès à des ressources disponibles sur Internet de manière simple et évolutive. Simple, ici, renvoie à l'accès transparent aux ressources.

1. Un ordinateur central ou *mainframes* est un ordinateur de haute performance utilisé à des fins de calcul à grande échelle qui nécessitent une plus grande disponibilité et de la sécurité. Historiquement, les mainframes ont été associés à une architecture centralisée plutôt que distribuée.

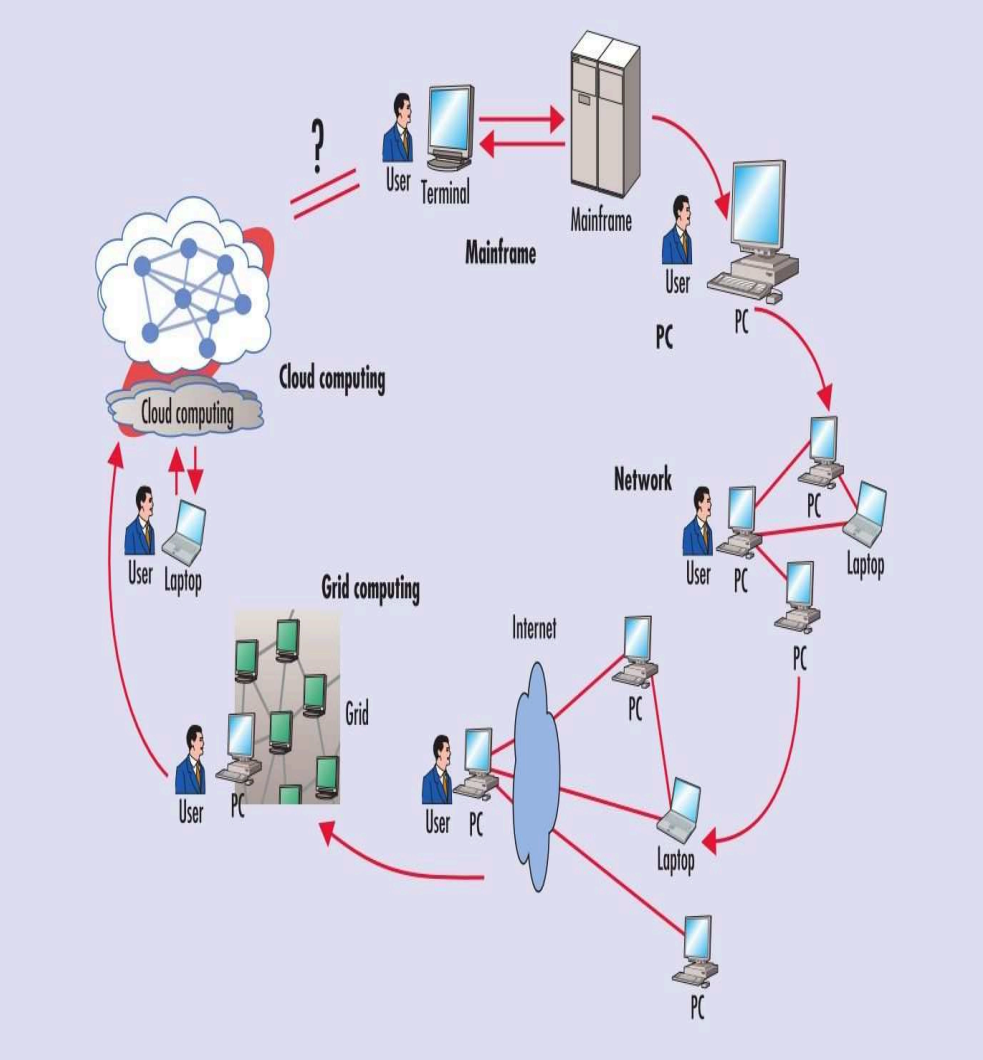


FIGURE 1.1 – *Les étapes de l'évolution de l'informatique selon [Voas 09]*

Et évolutive, parce que l'utilisateur peut se réapprovisionner en ressources selon ses besoins.

Bien que ces étapes apportent une compréhension de l'émergence du CC, nous constatons que sa définition n'est pas claire dans la littérature, ni les différences conceptuelles qu'il présente vis-à-vis des autres paradigmes connexes tels que les GC ou encore les *Clusters* qui demeurent ambigües. Nous proposons de lever ces ambiguïtés en proposant une définition du CC.

1.2 Clusters, Grid Computing et Cloud Computing : Definitions

1.2.1 Le point de vue industriel

Le CC fait référence à un paradigme qui est fondé sur l'idée de fournir des ressources informatiques (logiciels/matériels) virtualisées comme services, en utilisant Internet comme voie d'accès. Les ressources ainsi virtualisées peuvent être catégorisées selon trois types de services : *Software as a Service* (Saas), *Platform as a Service* (Paas) et *Infrastructure as a Service*

(Iaas). Les utilisateurs n'ont pas besoin d'investir dans des systèmes couteux quand ils ont un besoin ponctuel ou particulier, mais peuvent « acheter des services informatiques », à la demande : ils payent ainsi pour ce qu'ils ont réellement utilisé.

Le terme *Cloud Computing* est apparu pour la première fois en 2007 et fut lancé par IBM comme un nouveau modèle de calculs en réseau qui dépasse les limites des systèmes de l'époque, à savoir le manque d'évolutivité et de passage à l'échelle [Armonk 07]. Probablement pour cette raison, le CC est souvent confondu avec le GC et les *Clusters* [2] ou « grappe de serveurs ». Pourtant, ces trois concepts ne sont pas synonymes.

Les travaux de Pfister [Pfister 98], et Buyya et Marshed [Buyya 02] ont permis d'aboutir à la définition suivante des *clusters* :

Définition 1.2 *Un cluster est un type de système parallèle et distribué, qui se compose d'un ensemble d'ordinateurs autonomes et interconnectés, qui travaillent ensemble comme une seule ressource informatique intégrée.*

Buyya et al. [Buyya 09a] ont par la suite proposé également une définition du GC, qui se distingue des clusters par le caractère autonome des ressources concernées :

Définition 1.3 *Une grille est un type de système parallèle et distribué qui permet le partage, la sélection et l'agrégation de ressources autonomes et distribuées géographiquement, de manière dynamique et à la volée selon leurs disponibilités, capacités, performances, coûts et qualité de service. [Buyya 09a]*

Le terme *Cloud* remonte lui aux débuts de la conception des réseaux. À cette époque, tout comme aujourd'hui, le rôle d'un ingénieur réseau était de concevoir un réseau qui fonctionne correctement. Beaucoup de temps et d'énergie étaient dépensés pour cartographier le réseau et ainsi comprendre quels appareils y étaient connectés et comment ils y étaient connectés. Alors, le réseau créé constituait le domaine de ce qui était connu et contrôlé. Ces réseaux n'étaient pas isolés, et étaient connectés à d'autres réseaux ou à Internet : il fallait intégrer ces informations dans la cartographie. Cependant, les ingénieurs n'avaient pas toutes les informations sur ces réseaux connectés, puisqu'ils étaient détenus par des tierces-parties. Ils les ont alors représentés comme des « *nuages* » pour symboliser l'opacité que ces réseaux avait pour eux [Shor 11]. Le nuage cache ainsi à l'utilisateur le manque d'information et la complexité sous-jacente du réseau connecté.

Donner une définition exhaustive du terme « Cloud Computing » est complexe, car, selon les points de vues, les différentes parties prenantes mettent en évidence différents aspects. Des experts du domaine des *Technologies de l'Information et de la communication* (TIC) ont tenté de définir le CC produisant plusieurs définitions de ce même concept [Geelan 09]. En analysant ces définitions, nous constatons que chaque expert tente d'expliquer ce qu'est le CC en exposant ses avantages et ses inconvénients. Ces définitions se heurtent de plus à la subjectivité de chaque expert ; deux points de vue illustrent particulièrement cette subjectivité. Le premier est celui de Reuven Cohen [Geelan 09] qui affirme que « *faire une distinction entre Saas, Paas, Iaas est secondaire car en fin de compte toutes ces approches tentent de résoudre les mêmes problèmes (le passage à l'échelle). Comme les logiciels ont connu une transition depuis le modèle de déploiement*

2. Nous utilisons le terme *Cluster* en lieu et place de l'expression « Grappe de serveurs » tout au long de ce manuscrit.

en local vers celui en réseau, le Cloud sera l'élément clé permettant de développer, déployer et gérer des applications dans ce nouveau modèle. » ; tandis que Brian de Haaff [Geelan 09] considère que : "il y a beaucoup de confusion quand il s'agit de parler du CC, pourtant il n'est pas si compliqué. Il y a seulement trois types de services qui sont : SaaS, Paas et Iaas. ».

S'ils peuvent paraître contradictoires, ces deux points de vue révèlent des éléments intrinsèques du CC : le premier insiste sur l'élasticité et le passage à l'échelle qu'il permet, et le second prouve que cette capacité n'est pas la seule caractéristique du CC, son innovation principale étant le fait de proposer un modèle de service en couches. Enfin, dans la communauté industrielle, seul T-system propose une définition propre du CC [Kunesch 10] :

Définition 1.4 *Nous considérons le « Cloud Computing » comme la location d'infrastructures et de logiciels, ainsi que de bande passante, selon des conditions de service définies. Ces composants doivent pouvoir être ajustés quotidiennement en fonction des besoins du client et se caractériser par une disponibilité et une sécurité optimales. Les engagements de niveau de service (SLA) de bout en bout et la facturation sur l'usage sont inclus dans le « Cloud Computing ».*

1.2.2 Le point de vue de la recherche scientifique

Concernant les contributions de la communauté scientifique, nous avons effectué une recherche bibliographique sur le site *Science direct* pour la requête CC en filtrant sur les articles de types « État de l'art, compte rendu de lecture » entre 2008 et 2012. Les résultats obtenus indiquent 336 articles qui traitent de cette thématique. Parmi ces résultats, le premier chronologiquement à proposer une définition du CC est celui de [Buyya 09b] :

Définition 1.5 *Le Cloud est un type de système parallèle et distribué constitué d'un ensemble d'ordinateurs interconnectés et virtualisés présentés comme une ou plusieurs ressources informatiques unifiées, qui sont dynamiquement provisionnées selon une convention de service*[3] *établie entre le fournisseur de service et le consommateur.*

Le *National Institute of Standards and Technology* (NIST) a établi un groupe axé sur la promotion de l'utilisation efficace et sécurisée du cloud au sein du gouvernement et de l'industrie qui a proposé la même année (2009) une définition du CC qui a été mise à jour en 2011[4] :

Définition 1.6 *Le Cloud Computing (CC) est un modèle permettant un accès ubiquitaire, à la demande (on-demand) et via Internet, à un ensemble de ressources configurables (ex : réseau, serveurs, stockage, applications et services) qui peuvent être rapidement utilisées et libérées avec un minimum d'effort et de gestion de la part du fournisseur [Mell 09].*

Comme l'expose T-system [Kunesch 10], le point commun de toutes les définitions du CC est le suivant : derrière l'expression « Cloud Computing », concept théorique et aujourd'hui encore abstrait, se trouvent les applications, plates-formes et infrastructures à la demande,

3. Traduction de l'expression « Service-level agreement » selon les sites : http://www.baldassari-avocats.com/ et http://www.droit-technologie.org/ consultés en Avril 2015

4. L'ancienne version se trouvait à l'adresse : http://csrc.nist.gov/groups/SNS/cloud-computing/.

extensibles et standardisées comme des services mis à la disposition d'un utilisateur d'Internet. Les ressources telles que les applications, la capacité de calcul et de stockage ne sont pas physiquement disponibles chez un prestataire, mais disponibles sur Internet.
À première vue, le CC semble être une combinaison de clusters et des GC, mais en réalité il est bien plus large que cela. En effet, le CC est un modèle où tout est considéré comme un service qui est alors l'objet de négociations entre le fournisseur et le client.

1.3 Les modèles de service du Cloud computing

Le service est l'entité centrale du CC et fait l'objet de négociations entre le fournisseur et l'utilisateur. Par conséquent, les modèles de services sont les éléments fondamentaux du CC. Le NIST [Mell 09] propose une définition du service CC.

Définition 1.7 *Un service est considéré comme un service Cloud s'il respecte les caractéristiques suivantes :*

- *on-demand et self-service : la possibilité pour l'utilisateur final d'avoir accès au service rapidement ;*
- *accès via un réseau large (Internet) en utilisant des terminaux standards : ordinateur, PC, mobile, etc ;*
- *élasticité et rapidité de la fourniture : pour suivre la demande et le passage à l'échelle de l'utilisateur ;*
- *service faisant l'objet d'un SLA entre l'utilisateur et le fournisseur ;*
- *service mesuré car ce dernier doit respecter le SLA et peut être facturé selon l'utilisation.*

A partir de ces caractéristiques, trois catégories de services sont généralement utilisées : le software as a service, le platform as a service et l'infrastructure as a service.

1.3.1 *Software as a Service* (Saas)

Le service qu'un *Software as a Service* (Saas) fournit au consommateur est la possibilité d'utiliser les applications s'exécutant sur une infrastructure du Cloud, accessibles à partir de différents périphériques clients, par le biais d'un navigateur Web ou d'une interface de programmation (ex : un *Customer Relationship Management* (CRM)[5] en ligne) [Mell 09]. Les consommateurs de ce service n'ont pas le contrôle sur l'infrastructure qui est souvent basée sur un système multi-tenant où les applications des différents consommateurs sont organisées en un seul environnement logique afin de réaliser une économie d'échelle et une optimisation de la vitesse d'exécution, de la sécurité, de la disponibilité et de la reprise d'activités. Dans ce contexte, une application multi-tenants est une application qui permet aux utilisateurs de partager les mêmes ressources matérielles, tout en leur permettant de les configurer selon leurs besoins comme dans un environnement dédié [Bezemer 10]. Un « tenant » est alors l'entité organisationnelle qui loue une solution Saas multi-tenant. Typiquement, un groupe d'utilisateurs formant les parties prenantes dans l'organisation.
Le modèle Saas est parfois appelé modèle *Application Service Provider* (ASP) et est annoncé comme la nouvelle méthode de partage de logiciel [Choudhary 07, sia 01, Turner 03]. En effet, Bezemer et Zaidman [Bezemer 10] explique que les avantages du mode Saas sont :

5. Le CRM regroupe l'ensemble des dispositifs ou opérations de marketing ou de support ayant pour but d'optimiser la qualité de la relation client, de fidéliser et de maximiser le chiffre d'affaires ou la marge par client.

— la capacité de partager plusieurs ressources matérielles par une même application ;
— la capacité d'une haute configurabilité du logiciel ;
— l'approche architecturale dans laquelle les utilisateurs partagent une seule application et instance de base de données.

Caractéristiques du modèle Saas

Il est important de veiller à ce que les solutions en mode Saas vendeurs respectent la définition d'un service CC. Certaines caractéristiques définissant le Saas comprennent [Kepes 13] :

— un accès Web au logiciel ;
— le logiciel est délivré en mode "one-to-many" c'est à dire : en mode multi-tenant ;
— l'utilisateur n'est pas tenu de gérer les mises à jour des logiciels et l'application des correctifs ;
— le service est fourni avec une Interface de programmation (API) permettant l'intégration et la communication avec d'autres services.

Avantages et limites du Saas

Les solutions CC sont de plus en plus répandues et le CC reste un modèle pour offrir accès à un nombre de services dans de délais courts. Cependant, il reste judicieux de se demander s'il est nécessaire de migrer vers des solutions CC surtout pour les entreprises qui veulent investir dans de tel paradigme. [Kepes 13] a exposé quelques point à considérer avant d'adopter une solution CC. Ils préconisent ainsi de migrer vers une solution Saas dans les cas exposés ci-après.

— Les offres et propositions où les solutions sont indifférenciées comme par exemple les emails. En effet, les offres en hébergement des emails ne confèrent pas d'avantages concurrentiels car tous les fournisseurs, quand bien même qu'ils n'utilisent pas le même logiciel, proposent des solutions dont les fonctionnalités restent inchangées étant donné qu'elles respectent des exigences fondamentale pour pouvoir gérer sa messagerie.
— Les organisations qui doivent interagir de manière significative avec un environnement extérieur comme dans le cas des campagnes emailing.
— Les applications qui nécessitent un accès nomade ou mobile et omniprésent. Un exemple serait les logiciels de gestion de ventes mobiles utilisées par les commerciaux sans cesse en déplacement.
— Les logiciels qui ne seront utilisés que pour un besoin à court terme, comme dans le cadre de collaboration pour un projet spécifique.
— Les logiciels qui peuvent être considérablement sollicités (avec une hausse de la demande) comme les logiciels de facturation qui suivent le développement de l'organisation et qui peuvent être sollicité au fur et à mesure que le nombre de clients et de commandes augmentent.

Bien que le Saas soit un outil précieux comme mode de livraison de logiciels en ligne, il n'est cependant pas adapté à toutes les situations et à toutes les organisations. En effet, les mêmes auteurs [Kepes 13] citent quelques exemples où le Saas n'est pas préconisé :

— les applications qui nécessitent une grande vitesse de traitement de données en temps réel ;
— les applications dont la législation ou la réglementation ne permet pas l'externalisation de données ;

— les solutions existantes sur site qui remplissent les besoins de l'organisation.

Voici quelques exemples de fournisseurs SaaS : le Salesforce Customer Relationships Management (CRM) system [6], NetSuite [7], GMAIL [8].

1.3.2 *Platform as a Service* (Paas)

Le Paas est une plateforme de développement qui supporte la gestion du cycle de vie du logiciel. La capacité fournie au consommateur est de déployer sur l'infrastructure Cloud des applications crées ou acquises pour un client à l'aide des langages de programmation, des bibliothèques, des services et des outils supportés par le fournisseur. L'idée derrière le Paas est de fournir au développeur une plateforme qui contient tous les systèmes et environnements nécessaires au développement, au test, au déploiement et à l'hébergement d'applications complexes fournies comme un service [Rimal 10]. Ainsi, la différence entre le Saas et le Paas est que le Saas inclut les applications achevées tandis que le Paas offre une plateforme où sont hébergées des applications en cours de développement [Dillon 10a]. Les avantages d'un tel service sont que le développeur peut bénéficier de l'infrastructure sous-jacente pour développer des applications conséquentes et accélérer l'exécution de celles-ci. Rimal et al. [Rimal 10] expliquent que le Paas peut réduire considérablement le temps de développement et propose également des centaines de services facilement accessibles.

Caractéristiques du modèle Paas

Comme le Saas, le modèle Paas possède un nombre de caractéristiques qui permettent de le définir en tant que tel. Ainsi le Paas est caractérisé par :

— un ensemble de services permettant le développement, le test, le déploiement, l'hébergement et la maintenance des applications dans une plateforme intégrée ;
— des outils de création d'interfaces utilisateur avec la possibilité de tester et de déployer différents scénarios ;
— une architecture multi-tenants où divers utilisateurs utilisent simultanément la même plateforme de développement ;
— la possibilité de développer des applications dans un environnement évolutif ;
— l'intégration de services Web et de bases de données en utilisant des standards ;
— la mise à disposition des équipes de développement, des outils de planification de projet et de communication pour faciliter la collaboration sur des projets.

Rimal et al. [Rimal 10] classent les plate-formes en trois catégories :

— *Plate-forme orientée intégration* : c'est une plate-forme qui permet la réalisation du e-business et qui offre des outils facilitant les échanges entre les applications et les utilisateurs telle que Salesforce App exchange [9] par exemple.
— *Plate-forme orientée développement* : c'est une plate-forme qui fournit un environnement pour les développeurs, leur permettant de développer, tester et déployer leurs applications, comme Google App Engine [10] qui met à la disposition des utilisateurs des environnements Java et Python.

6. https://www.salesforce.com/saas/ consulté en Juillet 2015
7. http://www.netsuite.com/portal/home.shtml consulté en Juillet 2015
8. https://mail.google.com/ consulté en Juillet 2015
9. https://appexchange.salesforce.com consulté en Juillet 2015
10. https://cloud.google.com/appengine/ consulté en Juillet 2015

— *Plates-formes orientées infrastructure* : c'est une plate-forme qui fournit aux développeurs des avantages en termes de capacités de stockage et de passage à l'échelle en mettant à leur disposition une infrastructure évolutive comme : AWS Elastic Beanstalk [11].

Avantages et limites du Paas

Le Paas apporte donc des possibilités intéressantes, particulièrement dans les situations où plusieurs développeurs collaborent sur un projet ou lorsque d'autres acteurs externes ont besoin d'interagir avec le processus de développement. Cependant, ce modèle de service n'est pas conseillé pour le développement des applications qui nécessitent la reconfiguration et la modification de l'infrastructure sous-jacente. Par ailleurs, le Paas est complètement dépendant du fournisseur en termes de disponibilité, d'exploitation de la plateforme, d'interfaces et d'outils. [Kepes 13] explique que le Paas peut se révéler inadéquat lorsqu'il s'agit d'applications nécessitant une haute portabilité, ou si le langage propriétaire proposé par le vendeur peut impacter le processus de développement. En effet, le fournisseur peut proposer des services, qui peuvent être des interfaces ou langages qu'il a lui même développés, ce qui peut affecter la migration des applications depuis un Cloud vers un autre générant le phénomène du « *data lock-in* ». Pour pallier ce problème, des efforts ont été faits afin de développer des Paas dites "open" : *Open Platform as a Service* (OPaas). OPaas est une étape dans l'évolution du Paas dont le but est d'offrir un accès ouvert aux interfaces de programmation d'application (API) et aux standards. Voici quelques exemples de OPaas en ligne : Cloud Foundry développée par VMware, OpenShift par Red Hat, Cloudify, Stackato et WSO2 Stratus.

1.3.3 *Infrastructure as a Service* (Iaas)

Dans le modèle *Infrastructure as a Service* (Iaas), le service fourni est de s'approvisionner directement en ressources de stockage, réseaux, systèmes d'exploitation et autres ressources et environnements informatiques dans lesquels le consommateur est en mesure de déployer et exécuter diverses solutions. Ce service est généralement destiné aux administrateurs systèmes. Nicholas Carr [Carr 06] explique que le Iaas est « l'idée d'acheter des ressources (infrastructure) informatiques - ou un datacenter en entier - qui peuvent être réapprovisionnées on-demand, sous la forme d'un service pour répondre à vos besoins. En raison des progrès rapides de la virtualisation, l'automatisation, et des stratégies de paiement, le *Hardware as a Service* (Haas) est prêt à être déployé ».

La *virtualisation* fait ici référence à l'abstraction des ressources logiques loin de leurs ressources physiques sous-jacentes afin d'améliorer l'agilité et la flexibilité, de réduire les coûts et donc d'accroître la valeur de l'entreprise [Golden 08]. Par exemple, utiliser un hyperviseur [12] comme *Hyper-V* de Microsoft afin d'installer plusieurs machines virtuelles avec différents systèmes d'exploitation sur un même serveur physique est une méthode de virtualisation. Le but de la virtualisation est d'améliorer l'utilisation des ressources en fournissant une plate-forme d'exploitation unifiée et intégrée pour les utilisateurs et les applications basée sur l'agrégation de ressources hétérogènes et autonomes [Sahoo 10].

11. http://aws.amazon.com/fr/elasticbeanstalk/ consulté en Juillet 2015
12. Un hyperviseur est une plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps.

Caractéristiques du modèle Iaas

Comme le Saas et le Paas, le modèle Iaas est en évolution rapide de par les avantages qu'il offre. Néanmoins, pour qu'un modèle Iaas soit défini en tant que tel, il doit respecter certains critères qui sont :

— les ressources sont fournies en tant que service à l'utilisateur ;
— il permet la mise à l'échelle dynamique ;
— il possède un coût variable, selon la configuration demandée et le fournisseur, et est facturé sur un modèle de service public (comme pour l'électricité par exemple) ;
— une seule entité physique (matériel) est utilisée par plusieurs clients.

Le principe du concept d'Iaas est de permettre l'externalisation du parc informatique et ainsi bénéficier des avantages offerts par le fournisseur en termes de maintenance et d'entretien du matériel. En effet, le fournisseur de Iaas est responsable de la récupération et de la reprise du service en cas de panne, mais aussi de la gestion de l'énergie dans ses datacenters comme par exemple les stratégies de refroidissement des serveurs. Ce modèle de service peut être avantageux pour les *Petites et Moyennes Entreprises* (PME) car il leur permet de réduire les coûts d'infrastructure informatique, mais aussi de se concentrer sur le coeur de métiers en sous-traitant la gestion de l'informatique à un tiers.

Ibrahim et al. [Ibrahim 11] présentent d'ailleurs ce modèle comme le plus supportable, à la fois dans l'industrie et le milieu académique, car il présente une solution informatique convaincante avec une capacité prouvée pour réduire les coûts et améliorer l'efficacité des ressources. De même, Rimal et al.[Rimal 10] expliquent qu'outre la flexibilité, l'avantage de l'Iaas est la méthode de paiement basée sur l'utilisation car cela permet aux clients de payer pour ce qu'ils ont réellement utilisé. Les mêmes auteurs décrivent les exigences fondamentales de Iaas qui sont : l'utilisation des dernières technologies avec les dernières mises à jour, sur demande, en toute autonomie et multi-tenants avec isolation des différents clients.

Avantages et limites du Iaas

Le Iaas est avantageux dans des situations qui sont étroitement liées aux avantages apportés par le CC.

— Quand la demande est volatile. En effet, Les entreprises doivent se positionner sur des marchés de plus en plus volatiles. Les nouveaux produits connaissent des cycles toujours plus courts. Les produits existants sont dépassés de plus en plus rapidement et disparaissent. Cela oblige les entreprises à innover sur des périodes de plus en plus courtes. Les entreprises doivent à présent réagir à ces changements si elles souhaitent être performantes sur le marché. La pression augmente donc non seulement sur la gestion d'une entreprise mais également sur ses *Technologies de l'Information et de la communication* (TIC) qui doivent être capables de s'adapter avec rapidité et souplesse à de nouvelles circonstances dans tous les processus commerciaux supportés et élaborés par les TIC[CC2].
— Pour les nouvelles organisations qui n'ont pas les moyens ni le capital pour investir dans les TIC.
— Lorsque l'organisation se développe et croît rapidement et que l'infrastructure existante ne peut pas suivre le passage à l'échelle.
— Lorsque l'organisation est contrainte de limiter les dépenses en capital pour passer en dépenses de fonctionnement [13].

13. Les dépenses de fonctionnement sont celles qui reviennent régulièrement chaque année : rémunération du

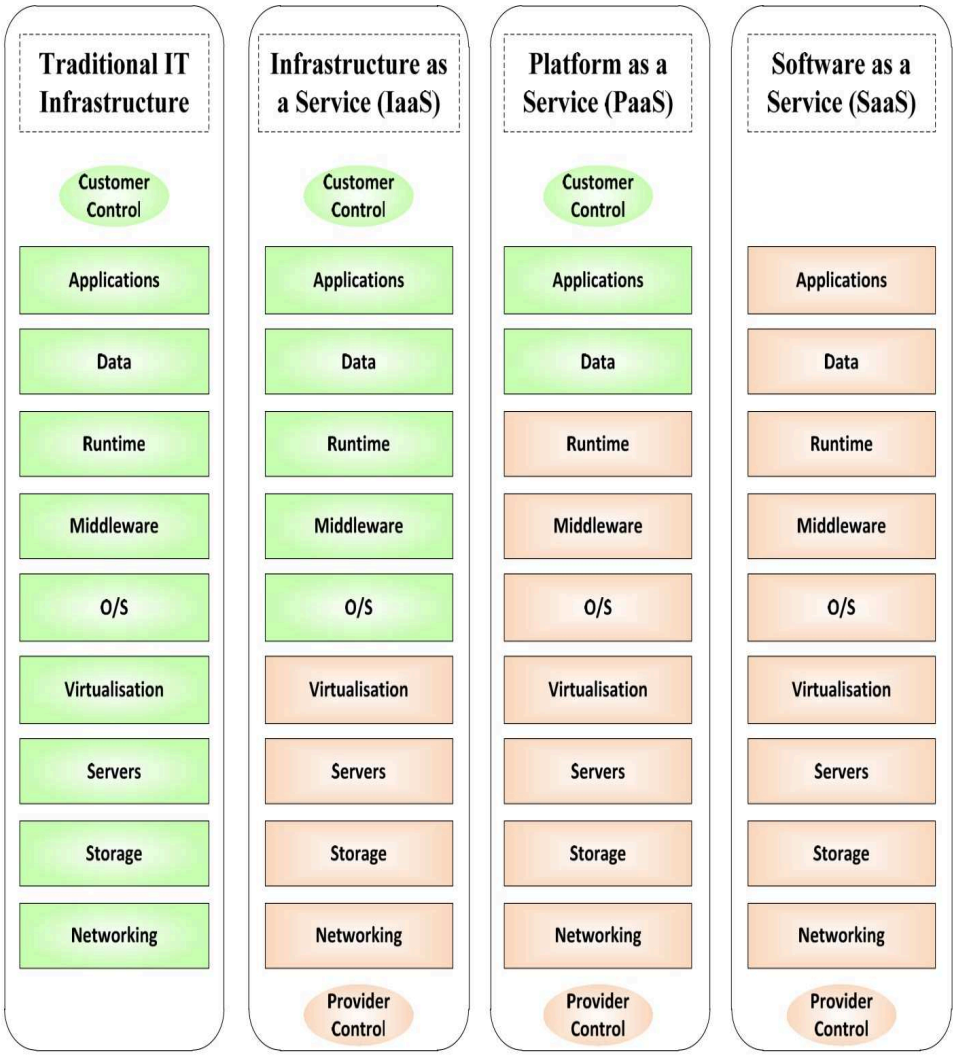


FIGURE 1.2 – *Les niveaux d'interaction entre l'utilisateur et le Cloud [Wang 14]*

— Pour une activité spécifique qui nécessite par exemple une infrastructure pour effectuer des tests ou pour répondre à des besoins temporaires.

Bien que le modèle Iaas offre d'importants avantages, ses limites peuvent s'avérer problématiques, notamment lorsque la conformité réglementaire de l'entreprise rend difficile la délocalisation ou l'externalisation des traitements et le stockage de données ; ou lorsque l'infrastructure locale répond aux besoins de l'organisation en termes de passage à l'échelle.

Cependant, comme le montre la figure 1.2, dans les différents modèles de services, le niveau d'interaction de l'utilisateur avec le Cloud est différent. En effet, l'implication de l'utilisation et son contrôle du système diminue quand il passe d'une architecture traditionnelle à un Cloud. Par conséquent, la gestion et la maintenance de l'infrastructure dépend de plus en plus du fournisseur de service [Wang 14]. RunAbove [14] et Amazon EC2 [15] sont des exemples de fournisseurs Iaas.

personnel, fournitures et consommation courante, petit entretien, participation aux charges d'organismes extérieurs, paiement des intérêts des emprunts et des dettes, et des frais financiers, etc.

14. https://www.runabove.com/index.xml consulté en Juillet 2015
15. http://aws.amazon.com/fr/ec2/ consulté en Juillet 2015

1.4 Modèles de déploiement du Cloud Computing

Il existe quatre modèles de déploiement de CC : Privé, Communautaire, Public et Hybride [Mell 09, Rimal 10, Dillon 10a]. Les modèles de déploiement caractérisent globalement la gestion et la stratégie adoptées pour la livraison des services décrits précédemment [Jansen 11].

1.4.1 Cloud Privé

Dans ce modèle, l'infrastructure cloud est provisionnée pour une utilisation exclusive par un organisme unique comprenant plusieurs consommateurs (par exemple les business units). L'idée derrière le cloud privé est de partager des services et infrastructures fournis par une organisation ou son prestataire de service spécifié dans un environnement mono-locataire. Il peut être détenu, géré et exploité par l'organisation, un tiers, ou une combinaison d'entre eux, et il peut exister dans ou en dehors des locaux de l'entreprise cliente. Il n'est pas aussi rentable que le cloud public, mais il est moins cher que d'acheter et maintenir un centre de données. De même, il donne aux entreprises un haut niveau de contrôle sur l'utilisation des ressources de CC.

En effet, les raisons du choix du déploiement d'un cloud privé pour une organisation sont de maximiser et d'optimiser l'utilisation des ressources existantes en interne ; de dépasser les problèmes de sécurité, y compris la protection des données ; et de minimiser les coûts de transfert de données vers le cloud. Les coûts de migration vers un cloud public restent encore conséquents. Mais la raison principale du choix d'un cloud privé reste la maîtrise des activités et applications critiques pour l'organisation qui restent ainsi derrière leur pare-feu. Enfin, un autre domaine d'intérêt pour le Cloud privé est la communauté universitaires où il est mis en place souvent à des fins de recherche et d'enseignement.

Les clouds publics peuvent accéder aux données et services sur le cloud privé si cela est autorisé par l'organisation qui détient ces données. Cette décision doit être anticipée lors du déploiement du Cloud afin de prévoir une couche permettant l'échange et la communication avec les applications internes. Cette couche doit pouvoir localiser l'application appropriée pour récupérer les données nécessaires en toute sécurité. Selon [Rimal 10] la meilleure façon de mettre en place un Cloud Privé est d'avoir une stratégie de déploiement qui permette de construire un Cloud en utilisant des ressources internes et externes. Ce modèle a l'avantage d'être adapté aux besoins de l'entreprise néanmoins il est moins réactif en cas de montée en charge et peut s'avérer coûteux à mettre en place.

1.4.2 Cloud Communautaire

L'infrastructure cloud est provisionnée pour une utilisation exclusive par une communauté de consommateurs spécifiques venant d'un ensemble d'organisations qui ont des préoccupations et intérêts communs (par exemple : des missions communes, des exigences de sécurité et de politique de protection des données). Il peut être détenu, géré et exploité par un ou plusieurs organismes de la communauté, un tiers, ou une combinaison des deux. Il peut être déployé dans ou en dehors des locaux de l'organisation.

1.4.3 Cloud Public

Le concept de partage des services et infrastructures fournis par un tiers hors du site (qui possède et gère l'infrastructure physique) dans un environnement multi-tenants, est généralement appelé cloud public. C'est la forme dominante de modèle actuel de déploiement du

CC. Dans un Cloud public, l'infrastructure est provisionnée pour une utilisation ouverte pour tous les types de clients, que ça soit une entreprise ou une personne. Il peut être détenu, géré et exploité par une entreprise, une université, une organisation gouvernementale, ou une combinaison d'entre eux. Il décrit le CC dans son sens initial, où les ressources sont dynamiquement provisionnées sur un modèle de libre-service sur Internet, via des applications Web et des services Web.

L'infrastructure se situe dans les locaux du fournisseur qui possède la pleine propriété du Cloud avec ses propres politiques, valeurs, profits et coûts. Généralement, les entreprises ne veulent pas déplacer leurs applications à mission critique et leur core-business dans le cloud public pour des raisons de sécurité et de contrôle. Beaucoup de services de cloud computing populaires sont des nuages publics, y compris Amazon EC2, S3, Google AppEngine, et SalesForce.com.

1.4.4 Cloud Hybride

L'infrastructure cloud dans ce modèle est une composition de deux ou plusieurs modèles de déploiement (privé, communautaire, ou public) qui restent des entités uniques, mais qui sont liés ensemble par une technologie normalisée ou propriétaire qui permet la portabilité des données et des applications (par exemple, un Cloud hybride permet d'équilibrer la charge entre plusieurs Clouds).

Il se compose d'un ensemble d'établissements internes ou externes. Ce modèle de déploiement offre une réduction de coûts, une évolutivité et un passage à l'échelle des infrastructures on-demand et de la sécurité. En effet, la combinaison de cloud hybride avec l'infrastructure traditionnelle/existante peut être une solution viable pour la majorité des entreprises qui veulent optimiser l'utilisation de leurs ressources et accroitre leur compétences et savoir-faire en externalisant les activités périphériques vers le Cloud et contrôler les activités de base sur site à travers le Cloud privé. Cependant, les Clouds hybrides introduisent une complexité qui est de déterminer la façon de distribuer les applications à travers à la fois un Cloud public et un Cloud privé. Si la quantité de données est petite par exemple, un Cloud hybride ne se justifie que si une grande quantité de données doit être transféré dans un Cloud public. De plus, parmi les questions que le que le Cloud Hybride soulève, il y a celles des standards et de la normalisation, ainsi que celle de l'interopérabilité inter-Cloud.

1.4.5 Cloud Virtuel Privé

Le concept du *Cloud Virtuel Privé* de l'anglais *Virtual Private Cloud* (VPC) est un modèle qui a émergé récemment [Cohen 08, Wood 09] comme un moyen de gestion des ressources informatiques de manière à ce que celles-ci semblent être dédiées à une seule organisation d'un point de vue logique alors que l'infrastructure sous-jacente peut provenir d'un fournisseur de Cloud public, de l'organisation elle-même ou d'une combinaison des deux. Ce type de modèle a été mis en place par Amazon Web Services (AWS) [16] pour offrir à ses clients une solution qui servira de pont sécurisé et transparent entre l'infrastructure existante de l'entreprise et le Cloud public Amazon.

Le VPC est un concept à la fois fondamental et transformationnel [Nick 10]. D'une part, il propose une abstraction des ressources publiques qui, combinées à des ressources internes, fournissent des fonctionnalités équivalentes destinées à la même organisation. D'autre part, le VPC offre à une organisation la possibilité d'intégrer le CC dans son infrastructure existante.

16. http://aws.amazon.com/fr/vpc/ consulté en Juillet 2015

En effet, une fois que l'organisation gère ses infrastructures existantes, elle peut étendre de façon transparente son domaine de gestion pour englober les ressources hébergées par le fournisseur de Cloud public et accessible via un réseau privé virtuel [17] (de l'anglais *Virtual Private Network* (VPN)).

Dillon [Dillon 10b] explique que, bien que le VPC puisse paraitre comme une combinaison entre le Cloud Privé et le Cloud Public car il utilise les ressources informatiques mises en commun pour le grand public, il est pratiquement privé pour deux raisons. Tout d'abord, l'interconnexion entre l'infrastructure locale (de l'organisation) et le Cloud est assurée par un réseau privé virtuel offrant l'avantage de la sécurité assurée par le Cloud Privé. Avec ce mode de fonctionnement, toutes les politiques de sécurité de l'entreprise sont appliquées à toute l'infrastructure, même celle qui se trouve dans le Cloud. D'autre part, avec AWS par exemple, Amazon dédie un ensemble de ressources isolées pour créer le VPC pour une organisation spécifique sans que celle-ci ait à payer à l'avance. En effet, les utilisateurs bénéficient toujours d'un modèle de facturation "pay-per-use" pour leurs ressources dédiées. Ainsi le VPC représente un équilibre parfait ente le contrôle de l'infrastructure offert par le Cloud Privé et la flexibilité et le passage à l'échelle du Cloud Public.

Interactions entre le modèle de service et le modèle de déploiement

Dillon [Dillon 10b] précise que le modèle de service est orthogonal au modèle de déploiement. Un SaaS pourrait être provisionné sur un Cloud public ou Cloud privé ou la combinaison des deux si par exemple l'application est installée dans un Cloud public tandis que l'un des modules est déployé et maintenu sur un Cloud Privé.

1.5 Exigences architecturales pour les systèmes Cloud Computing

Nous présentons dans cette sections un résumé des exigences architecturales d'un système CC (figure 1.3) tels que présentées dans les travaux de [Rimal 10], [Xu 12c] et [Nick 10]. Ces exigences sont catégorisées suivant trois points de vue : celui du fournisseur, celui de l'entreprise qui utilise ou héberge le Cloud et celui de l'utilisateur final, comme le montre la figure 1.3.

17. Un réseau privé virtuel est un réseau informatique privé qui relie d'autres réseaux distants et souvent géographiquement séparés via Internet.

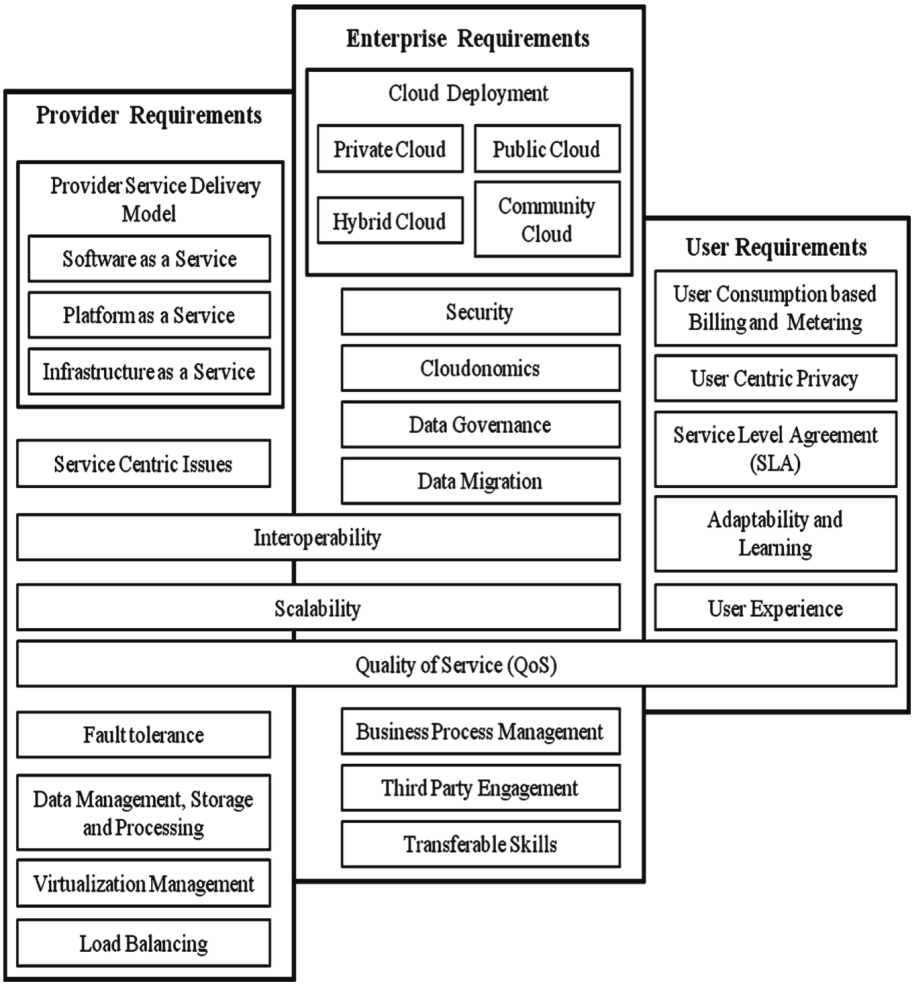


FIGURE 1.3 – *Exigences architecturales selon [Rimal 10]*

1.5.1 Les exigences du fournisseur

Du point de vue du fournisseur, le service doit répondre aux exigences des utilisateurs en terme de disponibilité, flexibilité et de support de la charge de travail. Ainsi, une architecture efficace est nécessaire afin de fournir des services virtualisés et dynamiques. En effet, selon J. Dean et S. Ghemawat [Dean 08], Google traite actuellement plus de 20 téraoctets de données brutes par jour. Ceci est un exemple qui montre le volume et la complexité de calcul qu'un fournisseur de Cloud doit traiter et gérer. Ce genre de charge augmente généralement la complexité de l'architecture et donc les coûts de fonctionnement d'un système de fournisseur de Cloud. Pour tenter de réduire la charge, les fournisseurs de services Cloud pourraient utiliser la discrimination des prix fondée sur plusieurs facteurs tels que les modèles d'utilisation, la bande passante et le niveau de service.

Exigence 1 : l'architecture support au modèle de service

Saas, Paas et Iaas sont essentiellement les trois types de prestation de services. Ces services sont habituellement fournis à travers des interfaces standards tel que les services Web, en utili-

sant l'architecture orientée services (*Service Oriented Architecture* (SOA)) ou le *Representational State Transfer* (REST).

Les architectures SOA sont un paradigme architecturale de plus en plus répandue dans le domaine logiciel [Gottschalk 00]. Il s'agit d'un ensemble de services pouvant communiquer entre eux, composé de fournisseurs de services et de consommateurs, qui utilisent les services publiés dans un dépôt ou via des courtiers [Alonso 04].

Définition 1.8 *Un courtier (Broker) est une entité qui gère l'utilisation, la performance et la prestation de services de CC, et négocie les relations entre les fournisseurs de Cloud et les consommateurs [Liu 11].*

Massuthe et al. [Massuthe 05] distinguent les fonctions des trois acteurs de cette architecture : un fournisseur de service (service provider) publie les informations relatives aux services disponibles dans un dépôt (repository) ; un demandeur de service (service requester) interroge le fournisseur de services pour répondre à son besoin ; un courtier ou intermédiaire (broker) de services permet au demandeur de services de trouver le service adéquat. Ils expliquent également le processus qui pilote les interactions entre ces trois acteurs : en fonction d'une requête du demandeur sous forme d'un service R, la tâche du courtier est de sélectionner parmi tous les services disponibles seulement les services P susceptibles d'intéresser le demandeur. Cette publication est appelée couche publique du SI ou couche blanche. Les deux services R et P ne doivent pas s'interbloquer (le blocage a lieu par exemple si R et P attendent mutuellement des informations l'un de l'autre ou envoient des messages imprévus).

Le modèle architectural sous-jacent au Web est appelé le Representational State Transfer (REST). REST répond au besoin de l'*Internet Engineering Task Force* (IETF) pour illustrer la façon dont le web fonctionne. D'après [Fielding 00] REST est un ensemble coordonné de contraintes architecturales qui tentent de minimiser la latence et la communication réseau tout en maximisant l'indépendance et l'évolutivité des implémentations des composants. REST permet la mise en cache et la réutilisation des interactions, la substituabilité dynamique des composants répondant ainsi aux besoins d'un système hypermédia (figure 1.4) distribué à l'échelle d'Internet.

Définition 1.9 *Un « hypermédia » [Tricot 95] est un ensemble de documents stockés sur support informatique, sous la forme de nœuds connectés par des liens. Il consiste à mettre en relation deux niveaux (Figure 1.4), le niveau des documents et le niveau de la base de données, le second niveau "organisant" le premier. Les nœuds sont constitués de textes, et/ou d'images, et/ou de sons, et/ou d'animations (vidéo par exemple). Chaque lien part d'un ancrage (mot, morceau d'image, zone d'écran, icône) dans le nœud d'origine, cet ancrage étant manifesté par un bouton (mot en gras, surligné, partie encadrée, icône). Le niveau "base de données" a un ensemble de caractéristiques, dont la plus importante est peut-être l'absence potentielle de contrainte :*

1. *sur les liens, il n'y a pas, a priori, de contrainte logique, ni sémantique, ni ensembliste,*
2. *sur les nœuds il n'y a pas, a priori, de contrainte de contenu, ni de taille.*

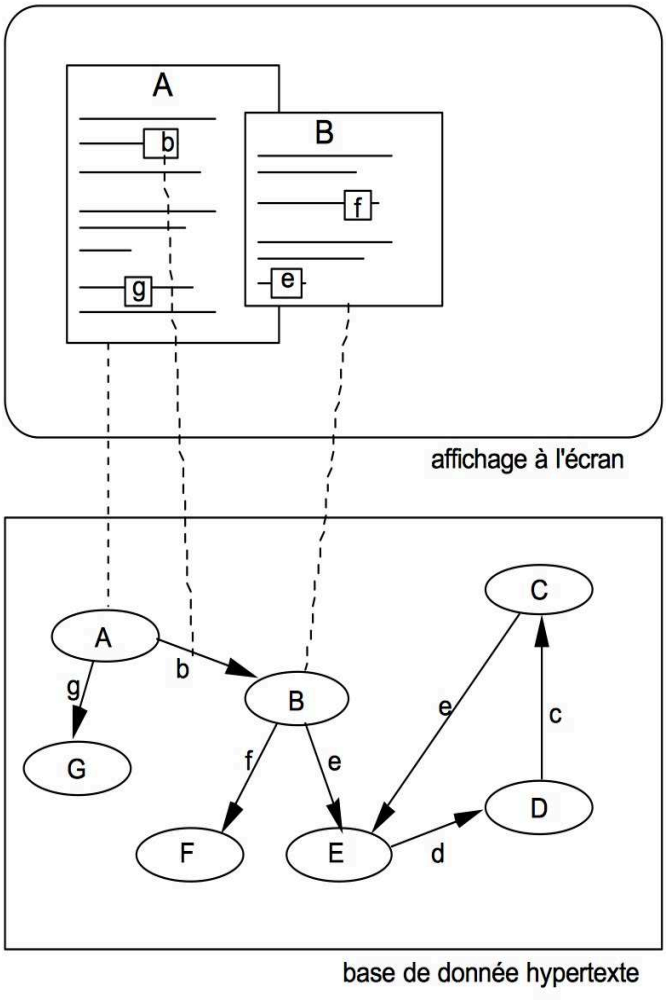


FIGURE 1.4 – *Représentation d'un système hypermedia [Conklin 87]*

[Richards 06] se basant sur la thèse de [Fielding 00] explique que REST distingue trois catégories d'éléments : les éléments de données, les éléments de connexion (connecteurs) et les composants aussi appelés les éléments de traitement.

Éléments de données. L'aspect clé de REST est l'état des éléments de données, les composants de REST communiquent via le transfert de représentations de l'état actuel ou souhaité des éléments de données. REST possède six éléments de données :

- Ressource (Resource) : Tout ce qui peut être nommé est une ressource. Une ressource est un concept représentant un ensemble d'entités mais n'est pas l'entité elle-même. Ainsi ce mappage conceptuel peut changer au fil du temps.
- Identifiant de ressource (Resource Identifier) : chaque ressource doit avoir un identifiant unique pour la distinguer des autres. Dans un environnement Web, les *Unified Resource Identifier* (URI) représentent les identifiants des ressources [Berners-Lee 04].
- Meta-données de ressource (Resource metadata) : Ceci décrit la ressource. Une méta-données fournit des informations additionnelles telles que des informations de localisations ou des identificateurs alternatives de la ressource.
- Représentation (Representation) : Tous les composants REST effectue des actions sur la représentation de la ressource. On n'envoie jamais une ressource et on ne la reçoit

pas, seulement la représentation de celle-ci est transférée entre les composants. Une représentation décrit l'état d'une ressource, par conséquent une ressource peut avoir de multiples représentations.
— Meta-données de représentation (Representation metadata) : pour décrire la représentation comme la dernière date de modification ou le type de média demandé.
— Données de contrôle (Control data) : Définit l'objectif du message échangé entre les composants, comme l'action demandée par exemple.

Connecteurs. Les connecteurs gèrent la communication entre les composants. Ils représentent les activités permettant l'accès à la ressource et le transfert des représentations. Les connecteurs dans *Representational State Transfer* (REST) sont de types :
— Client : envoie des requêtes, reçoit des réponses.
— Serveur : en écoute des requêtes, envoie des réponses.
— Cache : peut être du côté du client ou du serveur pour sauvegarder les réponses qui peuvent être mises dans le cache.
— Resolveur : transforme les URI en adresses réseau.
— Tunnel : permet de relayer les demandes, chaque composant peut passer d'un état actif pour occuper la fonction de tunnel.

Composant. Les composants REST sont identifiés par leur rôle dans une application et sont de types :
— Agent utilisateur : utilise un connecteur client pour initier une requête et devient le destinataire final de la réponse ;
— Serveur d'origine : utilise un connecteur serveur pour recevoir une requête et c'est la source définitive de représentations de ses ressources et il doit être le destinataire final de toute requête qui prévoit de modifier les ressources.
— Serveur mandataire (Proxy) : un composant mandataire est un intermédiaire choisi par un client pour fournir une encapsulation d'interface à des autres services, traduction de données, amélioration de performance ou protection de sécurité.
— Passerelle : un composant passerelle (c'est-à-dire un serveur mandataire inverse) est un intermédiaire imposé par le réseau ou le serveur d'origine pour fournir une encapsulation d'interface à d'autres services, pour la traduction de données, l'amélioration des performances ou l'application de la sécurité.

Il faut noter que la différence entre un serveur mandataire et une passerelle réside dans le fait qu'un client détermine quand il utilisera un serveur mandataire.

Exigence 2 : dépasser les problèmes engendrés par une approche centrée sur le service

Le CC en tant que service doit répondre aux exigences du monde de l'entreprise et ses besoins en TIC. Pour satisfaire ces exigences, une architecture CC doit faire face à une approche unifiée centrée sur le service. Ainsi, les services CC doivent être :
— Autonome : les systèmes et applications CC doivent être conçus pour s'adapter dynamiquement aux changements avec moins d'assistance humaine, cette autonomie peut être utilisée pour améliorer la qualité de service, la tolérance aux pannes ainsi que la sécurité ;
— Auto-descriptif : l'auto-description permet de représenter l'information contenue et les fonctionnalités du service de manière à ce que celles-ci soit réutilisables et indépendantes du contexte. Les services munis de cette capacité sont avantageux car ils

peuvent informer l'application cliente de la façon dont ils devraient être appelés et de quels types de données ils renvoient ;
— Une composition des applications distribuées à faible coût avec l'infrastructure nécessaire pour la collaboration et les interactions multiparties.

Globalement, le modèle centré sur le service doit inclure des processus liés au provisionnement et au déploiement ainsi qu'à la planification des services.

Exigence 3 : réaliser l'interopérabilité

[Wegner 96] définit l'interopérabilité comme :

Définition 1.10 *l'aptitude de deux systèmes (ou plus) à communiquer, coopérer et échanger des données et services, et ce malgré les différences dans les langages, les implémentations et les environnements d'exécution ou les modèles d'abstraction [Wegner 96].*

L'interopérabilité se réalise à plusieurs niveaux [EIF 04].
— Niveau technique : pour pouvoir échanger des informations, le niveau technique doit garantir l'infrastructure nécessaire afin que le transport des données d'un système à l'autre soit possible. Dans l'exemple de deux personnes en communication téléphonique, les téléphones, câbles et réseau téléphonique doivent être fonctionnels pour établir la communication et donc garantir l'interopérabilité au niveau technique.
— Niveau sémantique : il est nécessaire que les données échangées soient comprises par les systèmes. Ainsi, une donnée chargée de sens devient une information et peut être traitée par le système cible. [Tsuchiya 93] explique que : « *quand on donne un sens à une donnée à travers un cadre d'interprétation, elle devient une information* ». Dans l'exemple de la communication téléphonique, les deux personnes doivent être capables d'interpréter le sens du message passé, pour cela, il faut qu'elles partagent un même « cadre d'interprétation » (un même langage).
— Niveau organisationnel : ce niveau d'interopérabilité concerne les entités interagissant ensemble au sein d'une organisation. Par conséquent, une organisation adaptée doit être prévue pour assurer l'échange des informations. Dans le cas de la communication téléphonique, l'interopérabilité organisationnelle est assurée si aucune des deux personnes n'est absente.

Selon la définition donnée par [EIF 04], un système n'est interopérable que lorsqu'il respecte simultanément ces trois niveaux d'interopérabilité.

Dans le domaine du cloud, l'exigence d'interopérabilité implique la définition d'un cadre, d'une ontologie, d'un format de données, de protocoles et d'APIs ouverts qui permettent la migration et l'intégration des applications et données entre différents Clouds et facilitent également l'échange d'information entre les plates-formes. Cette exigence est essentielle à la fois pour les fournisseurs de service et pour les entreprises : par exemple, il faut assurer l'interopérabilité entre le Cloud et l'infrastructure existante de l'entreprise, mais aussi l'intégration de différents Clouds et Grids.

Exigence 4 : assurer la qualité de service

La qualité de service *Quality Of Service* (QoS) est l'idée que le taux de transmission, le taux d'erreur, la gestion du trafic réseau et son optimisation ainsi que d'autres caractéristiques peuvent être mesurées, améliorées et, dans une certaine mesure, garantis à l'avance. En

général, la QoS offre la garantie de performance et de disponibilité ainsi que d'autres qualités de service telles que la sécurité et la fiabilité. La QoS lie le fournisseur de service et l'utilisateur : alors le SLA joue un rôle clé dans l'établissement de l'accord entre les deux parties. [Buyya 09a] explique que de le mode d'utilisation des applications dans le monde réel varient avec le temps et ce de manière imprévisible. Par conséquent, ces applications ont des exigences différentes en termes de QoS en fonction des modes d'interaction des utilisateurs (en ligne/hors ligne).

Selon [Rimal 10] la question est de savoir quelles sont les exigences en termes de performance des applications et de services que le client envisage d'utiliser. En effet, dans le cas où ce dernier cherche une haute performance du service, il se peut que le fournisseur ne soit pas en mesure de satisfaire ce niveau de performance tout le temps en raison de la latence inhérente du réseau internet. Étant donné que les attentes des utilisateurs en terme de Qos resteront toujours élevées, il est important de définir le niveau de tolérance des entreprises quand leurs exigences ne sont pas remplies.

Exigence 5 : être tolérant aux pannes

La tolérance aux pannes est la capacité d'un système à fonctionner malgré la défaillance d'un de ses composants. Ce mode de fonctionnement nécessite en général l'isolation du composant défaillant et la disponibilité d'un mode de récupération du système pour revenir à un état stable.

Si les données ou applications critiques sont hébergées dans le Cloud, il est impératif que le fournisseur de service dispose d'un tel mécanisme pour éviter aux entreprises dépendantes du Cloud de perdre de l'argent à cause de l'interruption de l'activité [Rimal 10].

Selon [Gong 10], il y a principalement quatre endroits ou les pannes sont susceptibles de se produire : au niveau du fournisseur, entre les fournisseur (dans le cas de composition de services par exemple), entre le fournisseur et l'utilisateur, entre les utilisateurs. Si l'erreur se produit au niveau du fournisseur, les sauvegardes et réplications des systèmes remplaceront la partie défaillante. Quand la panne se produit entre les fournisseurs de service, la méthode universelle implique l'annulation de la transaction, et la redirection vers d'autres fournisseurs offrant ainsi une répartition de la charge. Les pannes au niveau de l'interface fournisseur/utilisateur peuvent avoir lieu pour différentes raisons : congestion du réseau, dysfonctionnement au niveau du navigateur, expiration du délai d'attente d'une réponse à la requête ou encore les attaques causés par les pirates (hackers).

Les algorithmes Byzantins ont été largement utilisés pour la gestion des composants défectueux car ils permettent de pallier le problème du comportement arbitraire des ressources [Castro 99].

Finalement, les utilisateurs ne se connectent pas seulement avec des fournisseurs, mais avec d'autres utilisateurs aussi afin de partager et échanger des données et des informations. Par conséquent, un accès non contrôlé d'un utilisateur à une ressource critique peut perturber le fonctionnement du système sous-jacent au Cloud en question. Il existe des mécanisme de sécurité au niveau du matériel et des systèmes d'exploitation permettant la gestion des droits d'accès et ainsi la protection des données et du système et le maintien de la confidentialité.

Exigence 6 : la virtualisation

Comme expliqué dans la section 1.3.3, la virtualisation tend à faire fonctionner plusieurs entités logiques (virtuelles) sur une même entité physique. Un processus de virtualisation de

base contient principalement deux tâches : la première consiste à créer un modèle de présentation de la ressource en analysant ses caractéristiques [Vichare 09, Kim 13, Chen 13, Ren 11], et la seconde à encapsuler les informations de la ressources dans un service en utilisant les méthodes et technologies des services web [Liu 12, Shi 07]. Les préoccupations du fournisseur de service à ce niveau sont de mettre en place des stratégies pour gérer un ensemble de machines virtuelles, hébergées sur un même système d'exploitation, les évaluer, les tester ainsi que les déployer au bon client tout en assurant l'isolation adéquate pour la protection des informations.

Il ne faut cependant pas confondre virtualisation et CC (figure 1.5), car le dernier consiste essentiellement à fournir des ressources informatiques partagées, à la demande via internet selon des modèles de services ; ce qui est justement réalisable grâce à la virtualisation. Que l'on soit ou non dans le Cloud, il est toujours possible de virtualiser son infrastructure.

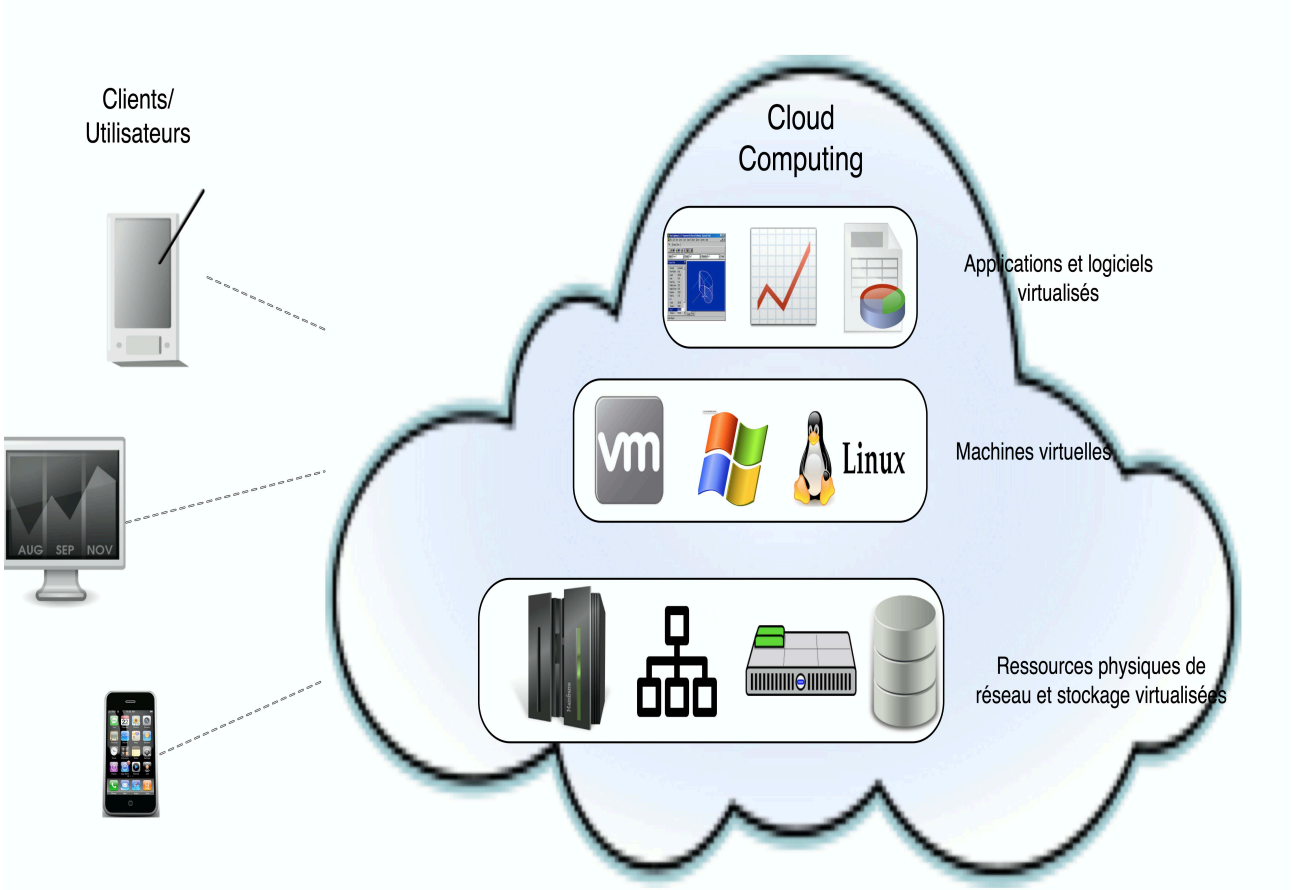


FIGURE 1.5 – *Le CC et la virtualisation*

Selon [Rimal 10], la virtualisation comprend :

Virtualisation de serveur : l'interprétation usuelle de la virtualisation des serveurs est le mappage d'une seule ressource physique vers plusieurs entités logiques ;

Virtualisation du bureau/client : la technique du client léger est l'un des modèles les moins coûteux pour réaliser la virtualisation à ce niveau ;

Virtualisation du stockage : fournit une abstraction entre le stockage logique et le stockage physique. il est par exemple possible d'effectuer plusieurs sauvegardes sur un même

disque physique. La capacité de stockage, la performance, la durabilité et la disponibilités sont des caractéristiques majeures à prendre en considération ;

Virtualisation du réseau : fournit un environnement pour exécuter plusieurs réseaux qui peuvent être personnalisés pour répondre à un besoin précis. C'est la combinaison des ressources matérielles et logicielles ainsi que les fonctionnalités du réseau en une seule entité logique ;

Virtualisation d'application : consiste en la virtualisation des applications et l'accès centralisé à ces applications via un serveur web. Ce niveau de virtualisation facilite l'octroi de licence et son provisionnement aussi bien qu'il permet la réduction des coûts ;

Virtualisation d'infrastructure : consiste en la séparation entre les applications et l'infrastructure ce qui permet aux développeurs de se concentrer sur le développement de l'application plutôt que de produire du code pour l'infrastructure ;

Virtualisation de ressource : l'idée derrière la virtualisaiton des ressources et le fait de personnaliser la ressource afin qu'elle réponde aux exigences des utilisateurs en termes de charge de travail. Une ressource physique peut être virtualisée en plusieurs ressources logiques. Les préoccupations à ce niveau de virtualisation sont la mise en place de stratégies pour la gestion des services et des ressources avec les politiques spécifiques à chacun.

Les mêmes auteurs précisent que la virtualisation est ainsi bien adaptée à une infrastructure Cloud dynamique car elle permet la mutualisation des ressources et offre des avantages importants en matière de gestion de systèmes complexes et d'isolation des données.

Exigence 7 : accompagner l'évolutivité du système

Nous utilisons le mot "évolutivité" pour exprimer la capacité du système CC à s'adapter aux changements pour supporter la charge de travail. Les travaux issus de la littérature scientifique utilisent le terme *scalability* pour exprimer cette capacité. [CC2] définit l'évolutivité comme :

Définition 1.11 *l'adaptation flexible et précise d'une solution matérielle ou logicielle aux exigences du consommateur [CC2].*

[Rimal 10] distingue deux types d'évolutivité : horizontale et verticale. L'évolutivité horizontale est ce que le CC offre grâce à des solutions d'équilibrage de charge et de livraison d'application comme les tables de hachage distribuées [18] permettant la répartition de la charge entre plusieurs nœuds afin de rendre le traitement des requêtes clientes plus rapide. L'évolutivité verticale quand à elle, est liée aux ressources utilisées comme dans le modèle des *mainframes*. Par exemple, une application qui dépend d'une base de données centralisée n'est pas évolutive verticalement à partir de l'instant où elle aura besoin d'une base de données supplémentaire et que ce besoin ne peut être satisfait. Selon l'auteur, les applications qui ne sont pas évolutives verticalement finissent par coûter plus lorsqu'elles sont déployées dans le Cloud en raison de l'accroissement des besoins en termes de ressources.

18. Selon [Galuba 09], une (DHT) est un système décentralisé qui fournit la fonctionnalité d'une table de hachage, à savoir, l'insertion et l'extraction de paires clé-valeur. Chaque nœud dans le système stocke une partie de la table de hachage. Les nœuds sont interconnectés au sein d'un réseau structuré, qui permet une réponse efficace aux requêtes d'extraction et d'insertion des clés et valeurs.

[Werner 06] explique que l'architecte doit prendre en compte l'axe vers lequel on veut faire évoluer le système, là où la redondance est nécessaire, et la manière dont le système gère l'hétérogénéité des ressources. Dans ce but, le rôle des architectes est primordial car concevoir et structurer une application évolutive pour une entreprise n'est pas une tâche triviale.

Exigence 8 : équilibrer la charge

L'équilibrage de charge (de l'anglais *load balancing*) est le mécanisme d'auto-régulation de la charge de travail au sein du CC [Rimal 10]. Il fournit une solution efficace pour des problèmes divers de mise en place et d'utilisation du CC comme par exemple : répondre à une charge de travail importante ou réduire l'indisponibilité potentielle du service. L'équilibrage de charge doit tenir compte de deux tâches principales : l'allocation de ressources et la planification de tâches dans un environnement distribué. Une allocation de ressources et une planification de tâches efficaces permettent d'assurer [Mayanka Katyal 13] :

— la disponibilité des ressources à la demande ;
— l'utilisation efficace des ressources selon la charge (élevée ou faible) ;
— l'économie d'énergie en cas de faible charge ;
— la réduction du coût d'utilisation des ressources .

L'équilibrage de charge est souvent utilisé pour implémenter le mécanisme de basculement. Le basculement se produit lorsqu'une allocation de ressources est limitée, ou lors d'une défaillance matérielle, une coupure d'alimentation ou une interruption du réseau. Alors, les composants du systèmes sont surveillés en permanence, et en cas de non-réponse d'une ressource, l'équilibreur de charge est informé et ne renvoie plus de demande vers cette ressource. Cette caractéristique du CC est héritée des Grids et constitue une exigence clé dans le déploiement de systèmes CC dynamiques [Rimal 10].

1.5.2 Les exigences de l'entreprise

Les entreprises doivent contrôler les services pour lesquels elles paient et porter une attention particulière aux questions concernant le niveau de service réalisé, la confidentialité, la conformité, la propriété et la mobilité de données [David 09]. Comme prévu par Gartner [Gartner 09], 20% des emails des entreprises étaient hébergés dans le Cloud avant la fin de l'année 2012. Dans ces e-mails, comme dans l'information portée dans le cloud en général, se trouvent des éléments clefs pour l'entreprise. Ainsi, tout passage au cloud doit s'accompagner de la mise en place d'un suivi rigoureux, portant sur les exigences présentées ci-après.

Exigence 1 : le choix du modèle de déploiement

Il existe 4 modèles de déploiement : Privé, Communautaire, Public et Hybride dont nous avons décrits les caractéristiques ainsi que les avantages et limites dans la section 1.4. Chacun de ces modes possède ses avantages et ses limites, qui fait du choix du modèle de déploiement pour l'entreprise un élément majeur de sa stratégie globale. [Rimal 10] explique que pour de nombreuses entreprises, adopter un Cloud privé pour déployer leurs technologies clés et créer leurs propres centres de données est un choix à privilégier par rapport à une migration vers un Cloud public.

Exigence 2 : vérifier la sécurité

La sécurité quand on parle de CC est la préoccupation centrale des utilisateurs. Les défis de la sécurisation des données, des applications et des infrastructures dans le Cloud sont considérables, particulièrement lorsqu'il s'agit de Clouds publics dont l'infrastructure et les ressources informatiques sont détenus par un tiers qui fournit des services au grand public par l'intermédiaire d'une plate-forme multi-locataire [Jansen 11]. En effet, les centres de données contiennent des informations qui sont traditionnellement stockées directement sur l'ordinateur de l'utilisateur final. [Jansen 11] résume les points qui nécessitent d'être considérés par les entreprises en termes de sécurité et de confidentialité avant d'adopter une solution Cloud :

— prévoir soigneusement les aspects de sécurité et de confidentialité des solutions CC avant de s'engager auprès des fournisseurs ;
— bien comprendre l'environnement des Clouds dont le mode de déploiement est public ;
— s'assurer que la solution Cloud répond aux exigences de sécurité et de confidentialité de l'organisation ;
— s'assurer que l'environnement informatique coté client répond aux exigences de sécurité et de confidentialité de l'entreprise pour adopter une solution Cloud ;
— prévoir la collecte régulière et l'analyse de données sur l'état des applications et du système déployés dans le Cloud afin de gérer les risques de sécurité et de violation de la vie privée des utilisateurs.

[Rimal 10] explique que la gestion des utilisateurs par authentification ou contrôle d'accès à base de rôles peut être une solution pour repousser les attaques des pirates informatiques. Les types d'authentification tels que les *single sign-on* basés sur le protocole SAML (Security Assertion Markup Language) [OASIS 04] et le OpenID [19]sont des solutions qui s'imposent aujourd'hui pour la gestion des problèmes d'authentification entre les utilisateurs et les applications Cloud.

Exigence 3 : s'inscrire dans une démarche Business Process Management (BPM)

Le Business process management consiste en une cartographie et un pilotage des processus métiers de l'organisation afin de les automatiser et de les optimiser prenant en compte l'interaction entre les organisations. Adapté au cloud, il fournit une structure d'entreprise et un ensemble de règles cohérentes permettant d'avoir une vue d'ensemble sur l'organisation. Ce concept peut être amélioré dans un environnement Cloud en offrant une plateforme pour les entreprises qui combine un mode Saas et une application BPM, un CRM ou un ERP [20] par exemple.

Quand l'entreprise adopte des solutions Cloud, le retour sur investissement (ROI pour Retuen Of Investment) est très important. Cependant, la méthode pour mesurer ce retour sur investissement reste un sujet à débattre [Rimal 10]. Le Open Group Cloud Computing Work Group [CCW 10] a lancé le projet Cloud Business Artifacts (CBA) afin d'identifier les problèmes liés à l'évaluation du ROI et ainsi expliquer comment il peut être mesuré :

— évaluer le prix et les coûts des services Cloud ;
— méthodes de financement des services Cloud ;
— la capacité et l'utilisation comme indicateur de performances ;

19. http://openid.net consulté en Juillet 2015
20. L'ERP vient de l'anglais « Enterprise Resource Planning ». Littéralement, ERP signifie donc : « Planification des ressources de l'entreprise » On utilise parfois dans le monde francophone la dénomination PGI : « Progiciel de gestion intégré » Quoiqu'il en soit, un ERP a pour principale définition « Outil informatisé de pilotage de l'entreprise » [Baglin 15]

— la gestion de risque ;
— coût global de propriété.

1.5.3 Les exigences de l'utilisateur

Les besoins de l'utilisateur final sont le troisième facteur clé pour l'adoption un système Cloud dans une entreprise. Le système offert par le fournisseur de Cloud doit être suffisamment fiable pour migrer les données sensibles. En effet, les utilisateurs ont besoin d'avoir l'assurance que leurs données et informations sont protégées contre les intrusions et les pertes et qu'elles sont disponibles en cas de besoin et accessibles partout ou ils peuvent se trouver.

Exigence 1 : une facturation basée sur la consommation

La facturation des utilisateurs dans un environnement Cloud est la même que celle utilisée pour les services publics tels que l'électricité ou l'eau et est fondée sur la consommation. La gestion des coûts est importante pour l'utilisateur afin de planifier et contrôler ses décisions et ainsi permet de vérifier le rapport « taux utilisation de la ressource/coût ». [Rimal 10] explique que les utilisateurs ont besoin de transparence quant à leur consommation et facturation. Par exemple, une utilisation fréquente des services Cloud n'est pas avantageuse quand il s'agit de réduire les coûts. Aussi un architecte de systèmes Cloud doit fournir cette visibilité aux utilisateurs. l'Activity-Based-Costing (ABC) [21] [Louth 09] est un outil qui permet à l'utilisateur d'évaluer le coût de son implémentation Cloud.

Exigence 2 : respecter la vie privée de l'utilisateur

L'une des principales préoccupations concernant le CC est liée au stockage des données sensibles des utilisateurs et des entreprises. Dans des environnements Cloud, tels que les centres de données, la vie privée des utilisateurs devient un enjeu majeur [Cavoukian 08]. Par conséquent, il faut aux fournisseurs Cloud gagner la confiance de leurs utilisateurs en proposant plus de solutions de protection des données. [Rimal 10] ajoute que de nombreux utilisateurs ont déjà été exposés à une certaine forme de CC à travers l'utilisation de service de courrier électronique (e-mail) comme par exemple Gmail, ou des applications issues du Web 2.0 comme Facebook. D'autre part, il existe aujourd'hui des technologies et solutions qui peuvent assurer l'intégrité et la confidentialité des données et ainsi protéger la vie privée des utilisateurs comme :

— la compression et le cryptage des données par l'utilisateur lors de leur stockage afin que le fournisseur de Cloud ne puisse pas s'en servir ;
— les réseaux locaux virtuels qui offrent un accès distant sécurisé ;
— les médiateurs de réseaux, comme les pare-feux et les filtres de paquets, qui renforcent la sécurité des communications.

L'auteur rajoute que ces technologies sont assez matures et peuvent être utilisées dans des environnements Cloud sans difficulté technique mais auront certainement besoin d'être ajustées pour répondre aux besoins de chaque situation.

21. La méthode ABC signifie Activity Based Costing. C'est un outil d'analyse des coûts par activité. Ce modèle de travail permet de diviser les différentes activités d'une entreprise commerciale pour en analyser les coûts et dégager la rentabilité de chaque référence.

Exigence 3 : le service Level Agreements (SLAs)

Le contrat mutuel entre le fournisseur et les utilisateurs est appelé SLA. C'est un accord qui prédéfinit les services à fournir. [Rimal 10] explique que de nombreux fournisseurs utilisent le SLA mais qu'ils ne protègent pas entièrement l'utilisateur contre les pannes. Certains points sont à soulever afin de mieux constituer un contrat SLA :

— qui et comment mesurer la prestation de service ?
— comment développer une méthode de suivi de la performance ?
— que se passe-t-il si le fournisseur ne parvient pas à fournir le service décrit dans le contrat ?
— quel sera le mécanisme pour changer le SLA au fil du temps ?
— quel sera le mécanisme de compensation si le prestataire de service ne respecte pas les éléments décrits dans le SLA ?

Exigene 4 : satisfaire l'expérience utilisateur UX

Selon la norme [ISO-9241-210 10] l'expérience utilisateur correspond aux réponses et aux perceptions d'une personne qui résultent de l'usage ou de l'anticipation de l'usage d'un produit, d'un service ou d'un système. Elle permet de fournir un aperçu des besoins de l'utilisateur final de manière à maximiser la convivialité des applications. La conception et le déploiement dirigés par l'expérience utilisateur constituent la prochaine étape logique dans l'évolution du CC. En effet, les applications se trouvant dans le Cloud doivent être faciles à utiliser, et surtout capables de fournir des services fiables qui sont facilement extensibles et personnalisables pour répondre aux exigences de la localisation et de la normalisation. Les interactions homme-machine et l'ergonomie sont des exemples de principes pouvant être utilisés pour concevoir des applications Cloud-UX.

1.6 Avantages du CC

L'avantage du CC est qu'il offre une grande capacité de calcul et de stockage, tout en assurant une évolutivité et une élasticité améliorées. En outre, avec une efficacité et une économie d'échelle [22], les services Cloud deviennent non seulement une solution moins coûteuse, mais aussi une solution efficace pour déployer des services TIC respectant l'environnement [Shawish 14]. Le CC se distingue des autres paradigmes grâce à [Wang 08] :

— L'approvisionnement des services à la demande : le CC fournit des ressources et services pour les utilisateurs à la demande. Les utilisateurs peuvent personnaliser leurs environnements informatiques, comme par exemple, l'installation des logiciels, la configuration réseau et où les utilisateurs possèdent généralement des privilèges administratifs ;
— La QoS : les environnements informatiques fournis par le CC peuvent garantir la qualité de service pour les utilisateurs, par exemple, la vitesse du processeur, la bande passante des entrées et sorties systèmes, la taille de la mémoire, etc. Dans le CC, les fournisseurs se basent sur le SLA pour garantir la QoS ;
— Un système autonome : le CC est un système autonome et il est géré de manière transparente pour les utilisateurs. Infrastructure, logiciels et données peuvent être reconfigurés, orchestrés et mutualisés automatiquement à l'intérieur du Cloud ;

22. L'économie d'échelle est une expression faisant référence à la baisse du coût d'un produit à l'unité grâce à l'augmentation de la quantité à produire.

— Evolutivité et flexibilité : sont les deux caractéristiques les plus importantes qui entrainent l'émergence du CC. Les services offerts par le CC peuvent passer à l'échelle sur la base de différents paramètres tels que la localisation géographique, la performance de l'infrastructure ou encore la configuration des logiciels. Les plates-formes CC doivent être flexibles pour s'adapter aux différents besoins d'un nombre grandissant d'utilisateurs.

1.7 Les enjeux du cloud computing

Le CC a offert un certain nombre d'avantages aux utilisateurs et organisations qui l'ont adopté, cependant, il y a encore un nombre de défis, qui sont actuellement relevés par les chercheurs du domaine [Leavitt 09] :

1. **Performance :** l'enjeu majeur de la performance arrive dans le cas de transactions et applications gérant un volume de données important. Aussi, les utilisateurs qui se trouvent géographiquement à une grande distance du fournisseur peuvent rencontrer des problèmes de latence et de retard ;
2. **Sécurité et confidentialité :** les entreprises sont toujours préoccupées par la sécurité de leur données quand elles utilisent le CC. En effet, les utilisateurs sont inquiets face aux attaques et vulnérabilités des systèmes quand leurs ressources et données critiques sont à l'extérieur de leurs locaux. Ainsi les fournisseurs de CC doivent suivre les pratiques standards de sécurité comme le paramétrage des pares-feux, délimitation de zones, segmentation du réseau, la détection d'intrusion, etc ;
3. **Contrôle :** certains départements informatiques sont préoccupés par le fait que le fournisseur de service Cloud a le contrôle total des plates-formes et généralement ne conçoivent pas des plates-formes spécifiques pour les entreprises et leurs pratiques commerciales ;
4. **Coût de la bande passante :** avec les CC, les entreprises économisent de l'argent et réduisent les coûts sur l'infrastructure matérielle et logicielle mais peuvent rencontrer une augmentation des frais de bande passante. Le coût de la bande passante peut être faible pour les petites applications orientées web, qui ne nécessitent pas le traitement d'un volume important de données, en revanche, il peut augmenter de manière significative quand le volume de données à gérer devient important ;
5. **Fiabilité :** le CC n'assure toujours pas une fiabilité et un fonctionnement sans interruption. Il y a eu des cas ou le service a été interrompu quelques heures néanmoins des travaux sont en cours pour établir des normes et définir des meilleurs pratiques pour assurer la fiabilité du service. Dans le domaine de la recherche, les laboratoires HP et Intel ont lancé des bancs d'essai distribués avec des installations en Asie, Europe et Amérique du nord avec l'objectif d'assister le développement et l'innovation pour le CC. IBM a lancé le "Research Cloud Computing" qui est un ensemble de ressources, accessibles à l'échelle mondiale, pour servir de support les processus métiers.
6. **Data-lockin :** une des préoccupations majeures des utilisateurs de Cloud est d'avoir des données bloquées chez un certain fournisseur. Pour [Germain 13] le lock-in existe lorsque le coût du changement de la plate-forme technologique d'un vendeur vers une autre est à ce point onéreux que le client est incapable de quitter les offres du vendeur. En effet, les utilisateurs peuvent vouloir déplacer leurs données et applications et ainsi quitter un fournisseur qui ne répond pas à leurs besoins et exigences. Cependant,

dans leurs formes et fonctionnements actuels, les infrastructures et plate-formes Cloud n'appliquent pas des méthodes standards pour stocker les données et applications des utilisateurs qui ne sont par conséquent pas portables depuis un Cloud vers un autre [Shawish 14].

1.8 Conclusion

Á l'ère où l'architecture orientée service (SOA) est en train de changer la manière dont les développeurs conçoivent, déploient et maintiennent leurs solutions, le CC est un modèle émergent où les ressources (logiciels/matériels) sont traitées comme un service. Ce chapitre a été consacré à la description du CC et de ses caractéristiques.

Nous avons présenté les différents paradigmes et étapes qui ont permis au CC de voir le jour : les *mainframes*, Internet, les *clusters* et le *Grid Computing*. Nous avons ensuite détaillé les modèles de service fournis dans un système CC : Saas, Paas et Iaas ainsi que leurs modes de déploiement, à savoir : Privé, Communautaire, Public et Hybride. Nous avons alors constaté les situations où il est avantageux de migrer vers l'un de ces modèles. Nous avons ensuite exposé les différentes technologies permettant la mise en place d'un système CC et nous avons exprimé les besoins de chaque entité : fournisseur, entreprise et utilisateur, quant aux solutions Cloud.

Nous avons montré que le CC offre beaucoup d'avantages aux entreprise en termes de réduction des coûts, d'adaptation et de passage à l'échelle selon la demande. Cependant, il reste encore quelques enjeux qui sont sujets de recherche aujourd'hui et qui démontrent les limites de ce système.

Dans le chapitre suivant nous allons étudier les systèmes d'information (SI) industriels en détail - qui sont le domaine d'application de cette recherche - afin d'analyser leurs particularités, en vue de leur migration possible dans le Cloud. Nous verrons notamment quelles sont les difficultés auxquels ces SI sont aujourd'hui confrontés, et qui rendent intéressante l'utilisation de solutions cloud qui permettent de s'adapter à la demande volatile qui caractérise ces systèmes au sein d'un environnement où la compétitivité ne cesse de s'accroître.

2

Le Product LifeCycle Management et les problèmes de collaboration

Résumé du chapitre

Nous avons présenté dans le chapitre précédent notre domaine d'étude qui est le Cloud computing en montrant son coté novateur et comment il s'avère bénéfique d'utiliser des solutions Cloud dans certains cas. Dans ce chapitre, nous introduisons les frontières du terrain que nous envisageons, et donc nous présentons notre domaine d'application qui est le Product LifeCycle Mangement (PLM). La section 2.1 présente la définition du PLM du point de vue d'un des plus connus cabinet de conseil dans le domaine CIMdata, confronté à celle retrouvée dans la littérature. Cette définition est suivie par l'exposition des phases du cycle de vie du produit ainsi que les système d'information qui servent de support pour ces phases. Ensuite, est présenté l'objectif d'une approche PLM dans la section 2.3 structuré autour des échelles de temps différents : court, moyen et long terme. L'objectif initial du PLM étant le développement collaboratif des produits, nous présentons dans la section 2.4 les modèles qui ont permis de réaliser de telle collaboration : les entreprises en réseaux. Nous présentons ensuite les problèmes rencontrés dans tels environnements ou la disponibilité de l'information du produit est primordiale dans la section2.5. Un nouveau paradigme, basé sur le Cloud Computing (CC) dont le but est d'offrir une solution agile pour le développement collaboratif des produits : Cloud Manufacturing (CM) est présenté dans la section 2.6. Enfin la section 2.8 conclut ce chapitre.

Sommaire

2.1 Le Product Lifecycle Management

Les entreprises manufacturières d'aujourd'hui ont besoin d'être agiles. Durant les années 80s, l'accent était mis sur l'automatisation, alors que le mot clé pour les entreprises aujourd'hui est « collaborer ». Ces tendances émergentes dans la stratégie des entreprises forment la configuration de la structure opérationnelle au sein et entre les entreprises et améliorent la façon d'orchestrer les processus métiers et de se concentrer sur les valeurs fondamentales au sein des entreprises [Rosén 10]. En effet, dans un environnement industriel en concurrence croissante, les entreprises sont à la recherche de moyens pour rester compétitives. C'est alors que le Product Lifecycle Management (PLM) s'est imposé comme une solution pour supporter la collaboration.
L'expression Product Lifecycle Mangement est habituellement traduite en langue française par Gestion du cycle de vie des produits. La grande majorité des entreprises industrielles du secteur manufacturier ont engagé (ou ont mis en place) des projets PLM [Paviot 10]. Les petites et Moyennes Entreprises (PME), fournisseurs ou sous-traitant des grands donneurs d'ordre, investissent aujourd'hui dans des projets de transition vers une démarche PLM [El Kadiri 09]. L'objectif de cette section est de définir le PLM et dresser un état de l'art permettant de présenter les modèles qui ont permis la collaboration entre les entreprises dans le cadre des projets PLM et aboutir ainsi aux problèmes qui en découlent.

2.1.1 PLM : Définition

Il existe différentes définitions du PLM dans la communauté scientifique, mais aussi industrielle et celle des éditeurs informatiques. Le but ici, n'étant pas de confronter toutes ces définitions, nous allons alors citer celles qui reviennent le plus souvent dans la littérature scientifique ou dans les rapports techniques. Pour plus de détails sur les autres définitions, [Paviot 10] présente dans sa thèse une analyse des points de vue des grands industriels impliqués dans le domaine, ainsi que des travaux scientifiques.
Du coté des industriels, la définition donnée par CIMData [1] qui est un cabinet de conseil dans le domaine du PLM, est celle qui revient souvent dans la littérature. Ainsi, CIMData définit le PLM comme :

Définition 2.1 *Une approche stratégique qui met en œuvre un ensemble cohérent de pratiques permettant de supporter la création collaborative ainsi que l'organisation, la diffusion et l'utilisation des informations relatives à la définition du produit au travers de l'entreprise étendue, de la conception à la fin de vie, et d'intégrer les hommes, les processus, les systèmes d'organisation et d'information.*

Ainsi, pour CIMData, le PLM n'implique pas seulement l'information numérique relative au produit, mais se place dans une vision stratégique permettant la création collaborative et la gestion de ces informations intégrant les hommes, les processus et les systèmes d'information. Dans la littérature scientifique, [Terzi 05] propose une définition du PLM qui s'articule suivant trois axes : stratégique, organisationnel et technique. Il définit le PLM par :

Définition 2.2 *Une approche intégrée qui utilise les technologies de l'information pour permettre la gestion collaborative des données numériques relatives au produit, au cours de toutes les phases de son cycle de vie. Ainsi le PLM implique :*

1. http://www.cimdata.com/en/resources/about-plm

— *un point de vue stratégique, selon lequel le produit est considéré comme le seul créateur de valeur pour l'entreprise ;*
— *la mise en œuvre d'une approche collaborative permettant la mise en commun de toutes les compétences de l'entreprise, distribuées parmi différents acteurs ;*
— *l'adoption d'un grand nombre de solutions informatiques et d'outils.*

Nous remarquons alors que les mots clés qui émanent de ces deux définitions sont : aspect stratégique du PLM, phases du cycle de vie et systèmes d'informations que nous abordons dans ce qui suit.

2.1.2 Phases du cycle de vie

Le cycle de de vie d'un produit comprend cinq phases principales, sous forme de cinq verbes d'action [Stark 11], [Dutta 05] : imaginer / définir / réaliser / maintenir / retirer et disposer. Ces cinq verbes correspondent aux états communément définis du produit : idée, conception, fabrication, usage et maintenance, démantèlement et recyclage [Fortineau 13a]. Durant la phase de l'imagination du produit, les exigences du marché sont déterminées et un concept du produit est réalisé. La phase de définition comprend la conception détaillé du produit, la planification du processus de fabrication et le développement d'un prototype. La production et l'entreposage ont lieu dans la phase de réalisation. Pendant la phase de maintenance ou utilisation, la fabricant ou le fournisseur sont responsables de l'entretien du produit. Finalement, lorsque le produit n'est plus en fonction, il est retiré et disposé ou recyclé [Lee 08].
Ces phases sont groupées par Stark [Stark 11] en trois catégories (figure 2.1) :

1. Début de vie (*Beginning Of Life* (BOL)) : qui regroupe l'idée, la conception et la fabrication ;
2. Milieu de vie(*Middle Of Life* (MOL)) : qui comprend l'usage et la maintenance ;
3. Fin de vie (*End Of Life* (EOL)) : qui couvre le démantèlement et le recyclage.

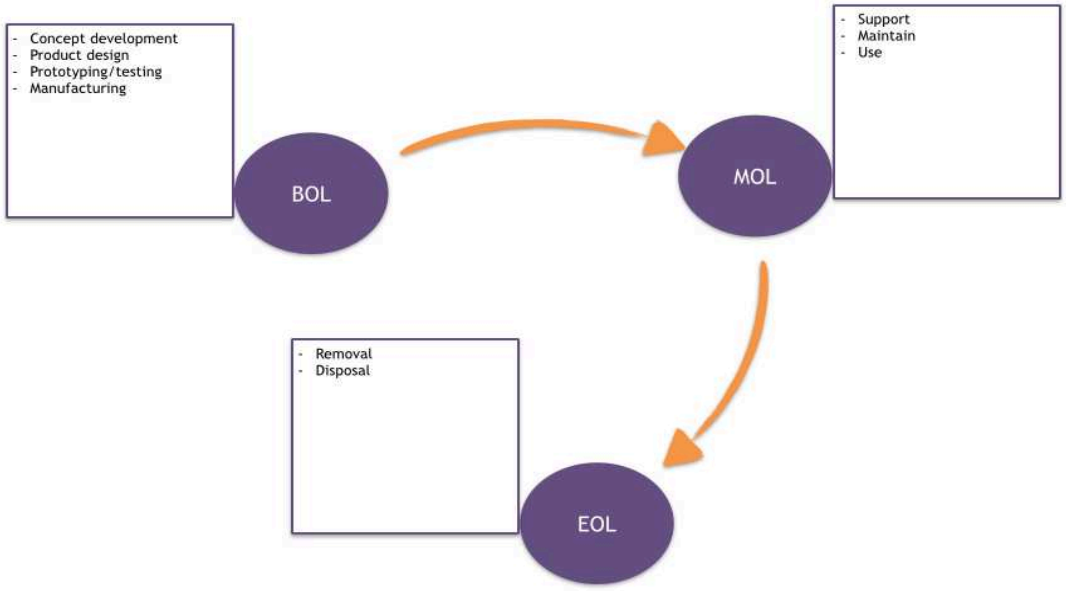


FIGURE 2.1 – *Les phases du PLM*

[Fortineau 13a] explique que les cinq étapes du cycle de vie délimitent en réalité des interfaces physiques qui sont le lieu de transformations majeures du produit, et par conséquent, elles délimitent des points de vue métier très différents.

Définition 2.3 *Le métier regroupe l'ensemble des savoirs et savoir-faire mobilisés par un opérateur pour mener à bien les activités relatives à un domaine identifié.*

Les interfaces entre les métiers impliqués dans le cycle de vie du produit sont les suivantes [Fortineau 13a], [Stark 11] :

— de l'idée à la conception du produit, il y a un changement d'acteurs : du marketing et/ou de la création vers les concepteurs. Mais surtout, il y a un changement de point de vue : on passe d'une description des besoins à celle de fonctions ;
— l'interface entre conception et fabrication est le passage du virtuel au réel. D'un produit unique et idéal, défini selon ses fonctionnalités, il faut passer à une somme de pièces, toutes différentes, bien réelles cette fois, qui, assemblées, donneront des produits eux aussi réels et variés, mais répondant aux exigences du cahier des charges. Le virtuel se transforme alors en réel, le fonctionnel en conforme ;
— entre le BOL et le MOL, c'est-à-dire de la livraison du produit conforme à son usage et sa maintenance, une seule différence, cependant majeure : l'apparition de la variable temps. Le produit s'use, se transforme, est modifié. Il passe de mains en mains et sort

généralement du périmètre de l'entreprise qui l'a fabriqué et/ou conçu. L'introduction du paramètre temps à cette étape a des conséquences importantes sur les paradigmes de modélisation de l'information liée au produit ;
— enfin, à l'heure du démantèlement, le produit perd son intégrité : il devient une somme de morceaux, regroupés non plus suivant des groupes fonctionnels, comme en conception, mais suivant des critères de recyclage : type de matériau, toxicité, biodégradabilité. Comment regrouper ses morceaux ? Comment en connaître la composition ? Tels sont les enjeux que posent aujourd'hui cette étape du cycle de vie.

Selon [Kiritsis 11] , le PLM ne considère pas assez les agents, ni la réutilisation de l'information car il ne ferme pas les boucles de l'information empêchant une réutilisation en amont (par exemple en conception) des informations de l'aval (par exemple de la maintenance).

2.1.3 Systèmes d'Information

Selon [Paviot 10] l'objectif du PLM est « *la maîtrise de la complexité qui caractérise le développement et le suivi des produits. Cet objectif s'inscrit dans une stratégie d'entreprise visant à réduire les coûts, les délais et à augmenter la qualité des produits* ».

[Simon 62] définit un système complexe comme un système constitué d'un grand nombre de composants qui interagissent d'une manière non-triviale. Dans de tels systèmes, compte tenu des propriétés des composants et des lois qui gouvernent leurs interactions, il est difficile d'en déduire les propriétés du système global. [Paviot 10] souligne que l'on a parfois tendance à confondre *complexe* et *compliqué* alors que ce n'est pas le « nombre » d'entités d'un système qui créent sa complexité mais plutôt la non-trivialité des interactions entre ces entités. Ainsi les systèmes d'information de l'industrie peuvent être considérés comme un système de systèmes [Zimmermann 08] et leur ensemble forme un système complexe[Paviot 10]. Selon [Rosén 10], une façon d'améliorer l'efficacité de la gestion de l'information est de traiter l'information numérique au sein des systèmes d'information (SI). Dans cette partie, nous présentons un récapitulatif non exhaustif des systèmes d'informations sous-jacents aux phases du cycle de vie.

Définition 2.4 *Un système d'information est un ensemble de ressources organisées pour la collecte, le stockage, le traitement, la maintenance, l'utilisation, le partage, la dissémination, la mise à disposition, l'affichage ou la transmission d'informations. Les ressources sont définies comme étant le personnel, les équipements, les budgets et les technologies de l'information (i.e. les ordinateurs et équipements connexes, les logiciels mais également le service - par exemple la maintenance - et les ressources associées). [CNSS 06]*

Le tableau 2.1 est extrait des travaux de [Fortineau 13a] qui dresse dans sa thèse une liste non-exhaustive de systèmes d'informations correspondants à chaque phase du cycle de vie. Elle explique que les informations concernant le produit peuvent se situer dans différents systèmes d'information selon la phase du cycle de vie. Par exemple dans la phase de conception, l'information peut se trouver dans les outils d'authoring, qui regroupent les SI de conception assistée par ordinateur (CAO) et de simulation, dans les SI de gestion des données techniques tels que les Product Data Management systems ou dans les outils de pliotage et de collaboration comme les MS Project.

Phase du cycle de vie	Type d'outils
conception	outil d'authoring (CFAO, simulation, bureautique)
	techniques (PDM)
	pilotage et collaboration (MS Project, plate-forme)
production	ERP
	MES
	APS
transport	TMS
distribution	DRP
	APS
	WMS
maintenance	GMAO
fin de vie	

TABLE 2.1 – *Exemple illustrant la diversité des systèmes d'information [Fortineau 13a]*

Dans la phase de production, sont cités trois types d'outils : les Entreprises Resource Planning (ERP), les Manufacturing Execution System (MES) et les Advanced Planning Systems (APS).

Définition 2.5 *Advanced Planning and Scheduling. L'APS est une application destinée à la planification. En fonction de la demande, elle permet d'analyser la capacité des ressources et les contraintes afin de proposer un horaire détaillé et adaptable pour une production optimale [Cardin 07, Genin 05a].*

Les Advanced Planning System (APS) permettent généralement de synchroniser les flux de la supply chain [Genin 05b]. Selon [Stadtler 05], les APS ont été développés pour combler les manques des ERP quant à la planification.

Définition 2.6 *Un ERP est un logiciel d'application personnalisable, standard qui comprend des solutions commerciales intégrées pour les processus de base (par exemple : la planification, le contrôle de la production, la gestion d'entrepôt, etc) et les principales fonctions administratives (comme : la comptabilité et la gestion des ressources humaines) d'une entreprise [Rosemann 99].*

Le MES vient compléter le Système d'Information d'Entreprise (SIE) en assurant des fonctions de contrôle/commande qui permettent le pilotage en temps réel des ateliers de fabrication [Huet 11].

Définition 2.7 *Un MES est un ensemble de composants matériels et logiciels permettant la gestion et l'optimisation des activités de production du lancement de la commande aux produits finis. Tout en maintenant des données précises et à jour, le MES guide, initie, répond et rapporte les évènements de*

l'usine au moment où ils se produisent. Un MES fournit des informations critiques aux modules d'aide à la décision à travers l'entreprise [Barkmeyer 99]

L'ensemble de ces applications offrent une vision à long terme de l'évolution du système. Elles constituent le niveau supérieur de décision dans le cadre CIM (Computer Integrated Manufacturing) illustré par la figure 2.2 où chaque niveau décide ce qu'un niveau inférieur exécute.

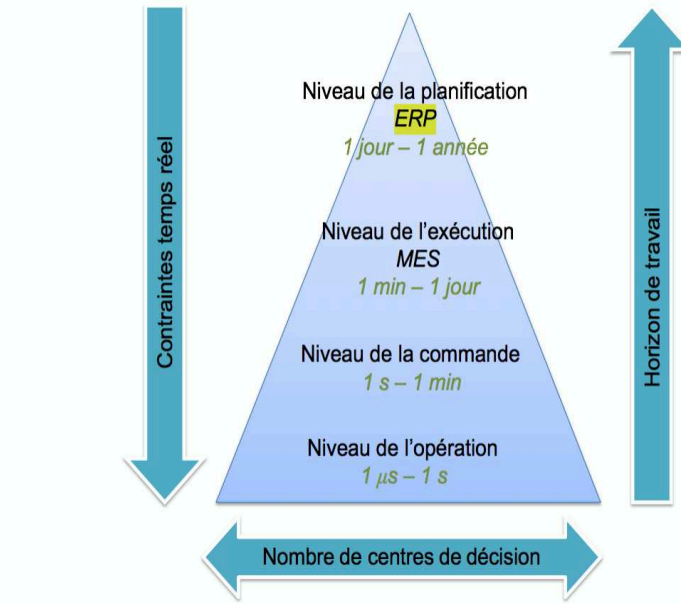


FIGURE 2.2 – *Les niveaux de décisions [Huet 11]*

Le niveau encore en dessous (la commande, niveau 1) comprend les applications et matériels nécessaires au contrôle/commande du système automatisé de production. Enfin, le niveau le plus bas (l'opération, niveau 0) contient tous les matériels nécessaires à la détection et a l'actuation : capteurs, actionneurs, etc [Huet 11].

Selon [Terzi 10], le PLM agit comme système de support pour les opérations concernant le produit dans le milieu de vie. Parmi les SI il y a les Transport Management Systems (TMS) pour le transport, les Distribution Resource Planning (DRP) pour la gestion des entrepôts et aussi des systèmes de Gestion de la Maintenance Assistée par Ordinateur (GMAO) pour la gestion des informations de maintenance.[Aberdeen_Group 07] explique que ces systèmes fournissent généralement des données de produits conjointement avec des capacités de gestion des informations de la maintenance des produits.

D'autres initiatives [Meyer 09] cherchent à mettre le produit au coeur des SI en lui faisant porter l'information pour le rendre actif et créer ainsi une sorte de produit « intelligent ». Nous présentons dans la section suivante ce type de systèmes.

2.2 Les SI contrôlés par le produit

Les progrès dans l'utilisation des technologies de l'information dans les systèmes de production donnent aux industriels une opportunité de promouvoir la personnalisation des produits. Cette personnalisation des produits avec l'utilisation de technologies telles que la RFID

ont fait émerger des paradigmes tels que les SCP et les HMS. Ces initiatives cherchent à distribuer des fonctionnalités classiquement réservées aux SI directement dans les produits et ainsi rendre actifs les produits en mouvement dans la chaîne logistique. Du point de vue des services offerts par l'objet dans le cadre d'une intelligence ambiante, on peut distinguer [Cea Ramirez 06] :

— L'objet porteur de données,
— L'objet pointeur vers un système d'information,
— L'objet fournisseur et demandeur de services,
— L'objet sensitif enrichi avec des capteurs et des actionneurs.

Le principe du concept SCP et de combiner de façon plus flexible des modes de pilotage centralisés avec des modes de pilotage distribués en tenant compte des capacités du produit à jouer un rôle actif de synchronisation des échanges entre différents systèmes d'entreprise de niveau « business » (ERP : Entreprise Resource Planning) et « process » (MES) [El Haouzi 08b] (figure 2.3) .

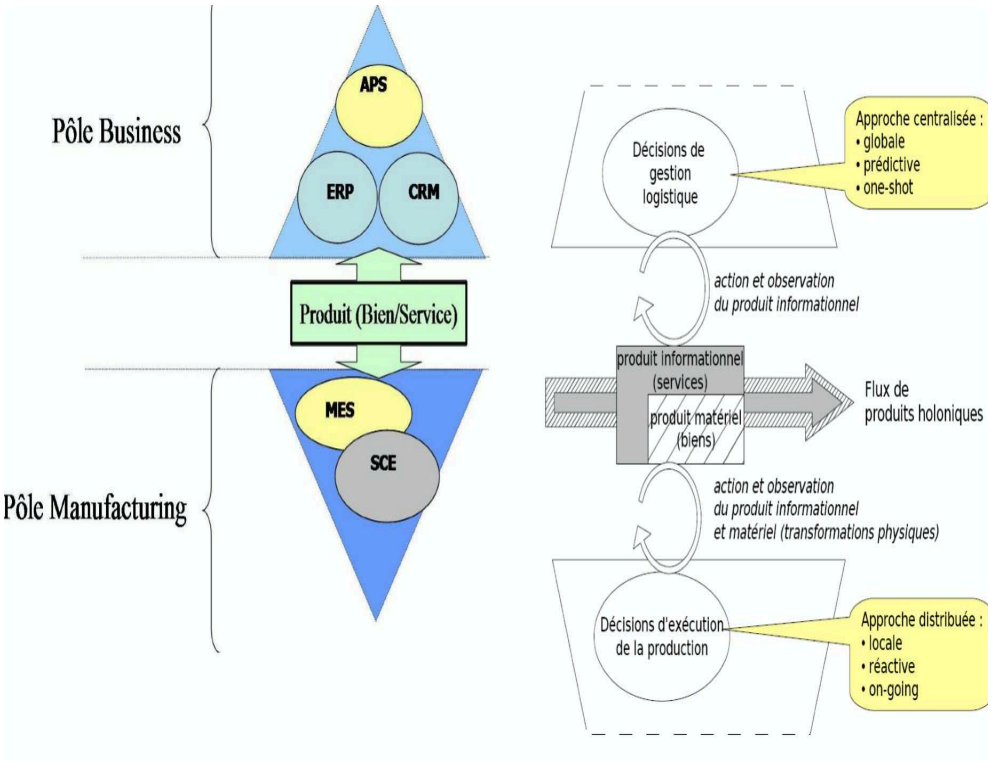


FIGURE 2.3 – *Séparation des pôles Manufacturing et Business [Baïna 06]*

La réalisation de tels systèmes a été l'objectif de la communauté Intelligent Manufacturing Systems (IMS) et l'a conduite à la définition et à l'expérimentation des Holonic Manufacturing Systems (HMS) . L'objectif de tels paradigmes est de rendre les systèmes intelligents et plus agiles. Le concept *Holon* a été introduit la première fois par Koestler dans son livre « The Ghost in the Machine » [Koestler 89]. C'est un néologisme combinant le mot grec *holos* qui représente « un tout », et du suffixe *on* qui, comme dans proton ou neutron, veut dire une particule ou partie. Du point de vue de l'auteur, un *Holon* représente à la fois une entité semi-autonome et une partie d'un tout. Les holons sont donc des systèmes ouverts et auto-régulés, qui présentent les capacités d'autonomie d'un tout (tendance à l'affirmation de soi), mais aussi les capacités d'obéissance d'une partie (tendance à l'intégration dans le tout) [Pannequin 07].

Dans [Brussel 98] les auteurs présentent le holon comme une entité autonome composée d'une partie informationnelle et une partie physique (Figure 2.4).

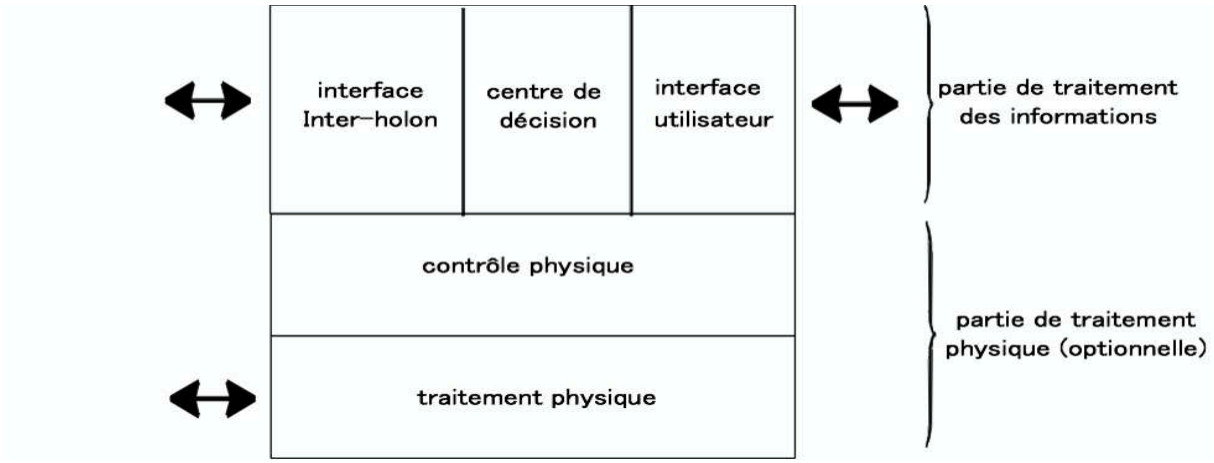


FIGURE 2.4 – *Architecture générale d'un holon [McFarlane 00]*

Quelques architectures ont été proposées dans la littérature pour modéliser des HMS, nous citons ADACOR (ADAptive holonic Control aRchitecture for distributed manufacturing systems) [Leitão 04] et HCBA (Holonic Component-Based Approach) [Chirn 00]. L'architecture de référence est PROSA qui a été définie pour modéliser et développer un HMS. Il s'agit de l'acronyme Product Resource Order Staff et comme son nom l'indique, l'architecture PROSA définit quatre types de holons.

- un Holon-Produit contient la connaissance nécessaire relative au produit et au procédé pour assurer la fabrication d'un produit avec une qualité correcte. Il a le rôle d'un serveur d'informations pour les autres holons du sysème. Il fournit des informations consistantes et à jour sur le cycle de vie du produit, les spécifications utilisateurs, sa gamme de fabrication et sa nomenclature ;
- un Holon-Ressource est constitué d'une partie physique (une ressource de production du HMS), et d'une partie de traitement de l'information qui contrôle cette ressource. Il offre une capacité de production aux holons alentours. Il contient des méthodes pour allouer des ressources de production, ainsi que la connaissance et les procédures pour organiser, utiliser et contrôler les ressources pour conduire la production. Un Holon-Ressource est une abstraction pour des moyens de production tels que des machines, des outils, stocks de matières, etc ;
- un Holon-Ordre représente un ordre de fabrication. C'est une entité active responsable de l'exécution correcte et à temps d'un travail. Il se charge explicitement des informations et du traitement des informations concernant un travail ;
- un Holon-Staff [Mařík 02] est une unité optionnelle qui permet la coordination entre les holons et d'assurer que l'objectif général de la production est atteint dont les ordres peuvent être rejetés par les autres holons.

Ainsi un système HMS peut être modélisé par l'ensemble de ces holons et les relations entre eux comme le montre la figure 2.5.

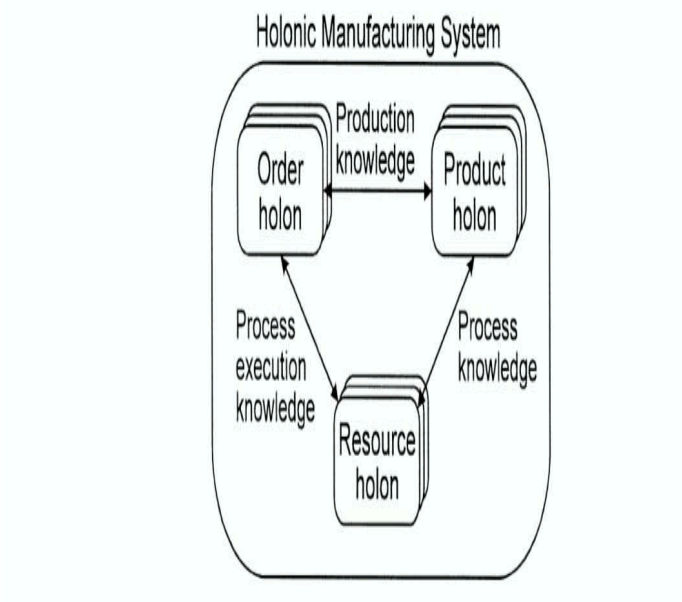


FIGURE 2.5 – *Bloc de construction basique d'un HMS [Brussel 98]*

2.3 OBJECTIF DU PLM

Devant les définitions proposées dans la littérature et présentées dans la section 2.1, [Paviot 10] a proposé dans sa thèse de définir le PLM selon ses objectifs et non pas selon ce qu'il est. Il précise ainsi que l'objectif du PLM est alors le suivant :

Définition 2.8 *L'objectif du Product Lifecycle Management est la maîtrise de la complexité qui caractérise le développement et le suivi des produits. Cet objectif s'inscrit dans une stratégie d'entreprise visant à réduire les coûts, les délais et à augmenter la qualité des produits [Paviot 10].*

Il structure ainsi les objectifs du PLM autour des échelles de temps différents : court, moyen et long terme [Paviot 10] :

- stratégique : l'ensemble des activités dont l'échéance est à long terme. On peut citer par exemple l'archivage des données qui décrivent le produit, la gestion de la traçabilité des décisions qui jalonnent le développement du produit, etc. ;
- tactique : l'ensemble des activités dont l'échéance se situe à moyen terme. Par exemple les processus liés à un projet particulier, l'analyse de leur robustesse, l'interconnexion de systèmes d'information de nouveaux partenaires etc. ;
- opérationnel : l'ensemble des activités qui s'inscrivent à court ou très court terme. Par exemple la planification des tâches quotidiennes des concepteurs, le pilotage d'atelier, la gestion des demandes de modification de conception (Engineering Change Request - ECR), la mise en œuvre de simulations numériques permettant de caractériser le comportement du produit virtuel etc.

De surcroît, le besoin initial du PLM était de permettre le développement simultané et collaboratif des produits en prenant en compte toutes les phases de leur cycle de vie [2] Ainsi

2. Cette affirmation est basée sur les définitions du PLM retrouvées dans la littérature comme la définition 2.2.

la réussite d ?une approche PLM nécessite que cette dernière joue le rôle d'un médiateur et assure la communication entre les SI utilisés par les différents partenaires du projet [Staisch 12]

Nous présentons dans la section suivante les modèles qui ont permis de réaliser la collaboration dans le PLM.

2.4 La collaboration dans le PLM

Dans un environnement concurrentiel où le développement du produit est trop complexe pour être traité par une seule entité, les entreprises établissent des partenariats avec d'autres organisations pour bénéficier de leurs compétences et infrastructures. En effet, les clients sont en quête de produits plus sophistiqués avec une meilleure qualité et des délais et prix réduits. L'entreprise étendue permet ce type de partenariat car elle représente une organisation de clients, fournisseurs et sous-traitants, engagés collectivement dans la conception, le développement, la production et la livraison du produit à l'utilisateur final, c-à-d. que ce modèle permet aux différents acteurs impliqués dans la réalisation du produit de collaborer tout au long du cycle de vie. [Browne 95] cite que ce modèle (l'entreprise étendue) permet de voir le système manufacturier comme un seul élément de la chaîne de valeurs qui fournit un service au client. [Jagdev 01] expliquent que le succès de l'entreprise étendue est dû à la vitesse et l'efficacité de l'échange et la propagation de l'information entre les différents partenaires dans un environnement collaboratif ; sous réserve que les participants aient des infrastructures IT et des outils de décision suffisamment sophistiqués pour rendre l'intégration possible et surtout transparente. Quand une entreprise établie un partenariat avec d'autres organisations indépendantes les unes des autres et distribuées sur une large zone géographique, le modèle résultant est appelé entreprise virtuelle [Zhao 99]. [Zhao 99] soutiennent que l'entreprise virtuelle permet à ses membres de répondre collectivement aux conditions du marché pour le développement de nouveaux produits dans des situations où les membres ne peuvent pas le faire individuellement. En effet, dans les entreprises virtuelles, les organisations partagent les coûts, compétences, expertise et ressources qui leur permettent d'accéder collectivement au marché mondial avec des solutions qui ne peuvent pas être fournies individuellement [Zhang 00]. L'entreprise étendue ainsi que l'entreprise virtuelle sont des variantes d'un modèle global qui est l'entreprise en réseau [Li 10b].

2.5 Les problèmes de collaboration dans le PLM

Pour une collaboration efficace dans les entreprises en réseau, les applications PLM ainsi que les informations du produit doivent être disponibles en temps et endroit voulus. Il est, par conséquent, nécessaire d'avoir une solution à la fois distribuée et robuste pour assurer la continuité du flux informationnel. Les solutions mises en place aujourd'hui ne sont malheureusement pas assez avancées pour pour répondre aux besoins grandissants des systèmes PLM actuels [Halpern 12]. [Elkadiri 08] mentionnent que le manque de flexibilité dans les systèmes PLM actuels forme une barrière pour une collaboration efficace.

Certains travaux proposent des solution basées sur les technologies Web dites "web-based manufacturing" pour rendre efficace la collaboration dans les entreprises en réseau. Sur le même principe, [Cutkosky 93, Cutkosky 98] proposent un portail pour la conception collaborative du produit basé sur l'integration de différrent outils de conception de manière à ce qu'une équipe puisse utiliser le Web comme un moyen pour partager les données du produit,

informations et connaissances durant cette phase. [Rodríguez 05], proposent une architecture pour gérer la connaissance pour le développement collaboratif du produit. L'architecture se compose de trois couches : information, application et enfin l'interface utilisateur et facilite le partage des connaissances nécessaire durant la phase de conception. Pareillement, [Janardanan 08] présentent un système qui offre une gestion collaborative de la structure du produit en utilisant le Web : visualisation et modification entre des concepteurs géographiquement distants. [Guerra-Zubiaga 06] argumentent que les structures de connaissance permettent le partage de l'expertise et augmentent la valeur ajoutée durant le travail collaboratif. Les structures de connaissance servent à stocker, gérer, classifier et utiliser les informations et connaissances pour faciliter le processus de conception.

[Zhang 14a] explique que les modèles actuels manquent (a) d'architecture ouverte et flexible, (b) de mécanismes efficace qui servira pour soutenir l'aspect dynamique de la collaboration, et (c) de solutions fiables et sécurisées. Par conséquent, les capcités offertes par les services actuels aux utilisateurs sont limitées et ainsi leur application peut être entravée.

Selon [Numata 98], des technologies Web peuvent être intégrées avec des systèmes de conception collaboratif pour le management efficace des données dans le cas de l'ingénierie des processus de développement de nouveaux produits.

Traitant l'aspect organisationnel de la collaboration dans le PLM, [Elkadiri 08] propose une méthodologie, basée sur des indicateurs pour le monitoring et le management des processus collaboratifs dans le PLM. Par ailleurs, PLMflow, proposé par [Zeng 02], est un système dynamique de workflow, qui est capable de supporter des processus non-deterministiques comme ceux trouvés dans les scénarios de conception collaborative, où les décisions prises par les partenaires nécessitent une composition dynamique ainsi qu'une modification du workflow en cours d'exécution.

Cet état de l'art révèle que les problématiques actuelles dans la collaboration des systèmes PLM sont liées à la flexibilité [Elkadiri 08] et le passage à l'échelle [Xu 12b]. En effet, les systèmes actuels ne sont pas capables de fournir des solutions raisonnables pour supporter le passage à l'échelle. Par ailleurs, il n'existe pas d'accès omniprésent aux équipements et ressources qui permet d'offrir une transparene aux différentes transactions et services. De plus, la majorité des solutions précédemment présentées se concentrent sur la phase de conception afin de permettre la collaboration entre différents acteurs ou traitent l'aspect organisationnel de la collaboration. Par conséquent, en plus d'être distribuées et robuste, l'architecture requise doit être flexible et doit supporter le passage à l'échelle des systèmes (Scalability en anglais) pour pallier les problèmes techniques et répondre aux besoins des entreprises.

2.6 L'émergence des solutions dites Clouds pour l'industrie

Pour pallier les problèmes de collaboration rencontrés dans le PLM des initiatives ont été menées pour rendre les systèmes plus flexibles et évolutifs pour suivre le changement au niveau des commandes et du marché en général. Des approches orientées web [Fuh 05], [Yang 03], [Wang 02] ont été proposées dans la littérature afin de répondre aux exigences du cycle de vie et ainsi faciliter la collaboration entre les acteurs. Nous avons présenté précédemment le CC et avons expliqué qu'il permet de réduire les coûts des infrastructures IT ce qui est bénéfique pour les petites et moyennes entreprises. En effet, contrairement aux grandes entreprises, les petites/moyennes entreprises ont besoin de techniques plus rentables pour gérer leurs activités et l'infrastructure sous-jacente en raison de leur budget plus réduit et de leur peu de connaissance dans les IT. De plus, elles sont constamment en croissance, et

leurs systèmes d'information peuvent évoluer selon leur passage à l'échelle, objectifs, modes de production et de gestion [Xu 08]. Selon [Ming 07], les nouvelles technologies doivent être développées pour permettre aux entreprises de livrer des produits adaptés aux besoins des clients avec un coût de production faible. [Ouzrout 12] estime que la prochaine génération de modèles et d'architectures doivent tenir compte des nouveaux modèles informatiques tels que les solutions à la demande et le CC. Le même auteur confirme que l'externalisation et l'utilisation des applications "en ligne" favorise l'agilité des entreprises, facilite l'échange et le partage de données, et permet de créer de nouveaux services de collaboration à faible coût. Basé sur le *Cloud Computing* (CC), récemment, un nouveau modèle orienté "Manufacturing" a été proposé : Le Cloud Manufacturing(CM) [Xu 12b], [Li 10a], [Tao 11]. Xu[Xu 12b] explique que le CM est une solution attrayante car le modèle manufacturier n'est plus orienté produit mais plutôt service. Le Cloud Manufacturing (CM) est un nouveau concept qui vise à apporter les avantages du CC à l'industrie manufacturière. En reflétant la définition NIST du CC, Xu [Xu 12b], définit le CM comme :

Définition 2.9 *un modèle permettant un accès ubiquitaire et à la demande à un ensemble de ressources manufacturières configurables et partagées. (ex : les logiciels de conception, les équipements de fabrication, etc) qui peuvent être rapidement provisionnées et libérées avec un minimum d'effort ou d'interaction de la part des fournisseurs de service [Xu 12b].*

Par conséquent, il faut une nouvelle architecture qui prend en charge les modèles manufacturiers de nouvelle génération et combler les besoins des systèmes actuels. Ceci nous permet de lever les verrous scientifiques quant à l'existence d'une architecture Cloud Manufacturing robuste, la robustesse étant l'habileté à maintenir l'efficacité à travers les différentes tâches, situations et conditions [Alberts 03], et ainsi à formuler notre problématique de recherche dans ce qui suit.

2.7 Problématique et démarche de recherche

2.7.1 Pallier les problèmes de collaboration dans le PLM

Comme dit précédemment, dans un contexte de mondialisation et de compétitivité croissante, les entreprises ont besoin d'outils et des méthodes leur permettant de réduire les coûts et délais de développements des produits. Ainsi, les entreprises doivent orchestrer de manière efficace toutes les informations et les opérations liées au produit tout au long du cycle de vie. Par conséquent, un accès ubiquitaire à l'information et aux ressources manufacturières est primordial afin de rendre celle-ci disponibles aux utilisateurs. [Xu 12b] explique que les systèmes manufacturiers en réseau manquent de plate-formes permettant la gestion centralisée des opérations mais offrant aussi la possibilité de choisir entre différents modes et services. Selon [Li 10a] et [F. Tao 10], les modèles manufacturiers en réseau existants font référence à un ensemble de ressources distribuées et intégrées dans le but d'entreprendre une seule tâche. Par conséquent, il y a besoin d'une plate-forme distribuée qui offre une plus grande flexibilité pour déployer des systèmes PLM. En effet, il est important, dans un environnement concurrentiel, qu'une organisation puisse choisir le prestataire qui est en mesure de lui offrir le service adéquat, et aussi d'intégrer ce dernier de manière rapide et transparente. [He 14] rapportent que les systèmes manufacturiers en réseau sont rigides du fait de leur incapacité à gérer un nombre variants de ressources manufacturière. Pour ces auteurs : les systèmes actuels

manquent de structures et de procédures adaptatives, qui seraient capables de supporter une utilisation et allocation dynamiques, flexibles et à la demande des ressources manufacturières. Par ailleurs, [Bussmann 99] soulignent que la « volatilité » de la demande oblige les fournisseurs d'adapter leur production aux exigences du marché, ainsi les systèmes manufacturiers doivent être capables de varier leur services selon la demande. Ceci implique que les systèmes doivent être évolutifs et capables de gérer le passage à l'échelle. L'évolutivité peut être réalisée soit par l'extension du temps de travail ou par l'ajout davantage de ressources pour suivre la charge de travail. Comme il est impossible de prolonger le temps de travail, ne pouvant pas dépasser dans le meilleur des cas 24h par jour, le meilleur moyen pour outrepasser cette limite est de rajouter des ressources (matérielles/logicielles).

Comme indiqué ci-dessus, un système de CM vise à apporter une solution en termes de mise à l'échelle suivant la demande, de flexibilité dans le déploiement et la personnalisation de solutions. Par conséquent, CM répond aux exigences des systèmes actuelles définies par [Molina 07] :

— intégration de l'entreprise et l'interopérabilité ;
— organisation distribuée ;
— environnements hétérogènes ;
— coopération ;
— agilité , évolutivité et tolérance aux pannes.

2.7.2 Problématique de recherche

Dans ce travail, nous proposons de partir de l'hypothèse que le Cloud Manufacturing peut être une solution aux problèmes de collaboration dans le PLM car il répond aux exigences des systèmes actuels présenté juste avant, pour évaluer d'une part à travers l'étude de la littérature les apports potentiels d'un tel paradigme aux systèmes industriels. Ainsi, nous pouvons formuler notre problématique comme :

« comprendre si et comment le Cloud Manufacturing, en tant que nouveau paradigme de livraison de services sur internet permet de dépasser les limites des systèmes actuels et d'assurer une collaboration efficace entre les utilisateurs et fournisseurs de solutions PLM tout au long du cycle de vie. »

2.7.3 Questions de recherche

Cette problématique de recherche implique ainsi de **définir une méthodologie qui permet de mettre en place une plate-forme Cloud Manufacturing et ainsi servir de support aux systèmes industriels traditionnels pour migrer vers le Cloud**.

Répondre à la problématique de recherche revient à répondre à deux questions de recherche sous-jacentes :

Comment modéliser notre domaine qui est le Cloud Manufacturing ?

Cette première question implique de définir un modèle générique permettant d'intégrer les différents fournisseurs de services, les utilisateurs, les services, les ressources et tout autre entité liée au Cloud Manufacturing. Ce modèle doit permettre d'intégrer ces entités malgré l'hétérogénéité des termes qui les décrivent.

Une fois ce modèle générique posé, il nous est possible de répondre à la question suivante :

Comment implémenter une plate-forme Cloud Manufacturing ?

Il s'agit ici de définir et implémenter une plate-forme Cloud Manufacturing permettant de mettre en correspondance des utilisateurs et fournisseurs dans le cadre de projets PLM.

La démarche de recherche envisagée afin de répondre aux différentes questions citées ci-dessus est la suivante :

Un état de l'art est réalisé en premier lieu. L'état de l'art s'articule autour de quatre axes majeurs :

— comprendre le Cloud Manufacturing et ses propriétés, et de vérifier l'existence d'une architecture Cloud Manufacturing dans les travaux issus de la littérature scientifique ;
— l'existence de cadre méthodologique permettant de mettre en place une plate-forme Cloud Manufacturing ;
— quel outil choisir pour modéliser le Cloud Manufacturing et les informations liées aux différentes entités ;
— quels outils et frameworks utiliser pour définir une architecture pour le Cloud Manufacturing.

Ensuite, nous proposons une méthodologie de mise en place d'une plate-forme Cloud Manufacturing. Cette méthodologie comprend deux étapes majeurs qui sont la définition d'un modèle générique de connaissance représentant le Cloud Manufacturing et d'une bibliothèque de composants informatiques qui, basé sur le modèle générique, fournit un ensemble de module représentant l'architecture d'une plate-forme Cloud Manufacturing.

Pour cette dernière étape, nous utilisons des outils de simulations d'environnements CC que nous adapterons à un environnement Cloud Manufacturing en implémentant les parties nécessaires.

2.8 Conclusion

Face à la compétitivité croissante, les entreprises sont à la recherche de nouvelles méthodes et outils permettant l'accélération et le développement de produit, assurant ainsi des gains en terme de coût de temps de développement. Le PLM s'est imposé comme une approche permettant d'atteindre ces objectifs tant il permet la gestion collaborative des données numériques relatives au produit au cours de toutes les phases de son cycle de vie. Nous avons présenté dans ce chapitre ce qu'est le PLM ainsi que les phases du cycle de vie d'un produit les regroupant dans trois catégories : BOL, MOL et EOL. Nous avons ensuite introduit les systèmes d'information sous-jacents à ces phases de cycle de vie qui permettent de gérer l'information numérique relative au produit. La notion de SI contrôlés par le produit, est abordée par la suite. Ces systèmes cherchent à mettre le produit au cœur des SI en lui faisant porter et capter l'information rendant l'environnement de développement ainsi plus « intelligent ». En nous basant sur les travaux de [Paviot 10], nous avons expliqué les objectifs du PLM en structurant ceux-ci autour de trois échelles de temps : court, moyen et long terme. Nous avons expliqué ce qu'est la collaboration dans le PLM et les modèles qui ont permis ceci : les entreprises en réseaux. Nous avons montré par la suite qu'une collaboration efficace nécessite que l'information liée au produit doit être disponibles en temps et endroit voulus, et avons détaillé les problèmes de collaboration annoncé dans la littérature. Le manque de flexibilité, de passage à l'échelle, d'accès omniprésent à l'information et à la ressource ainsi que la rigidité quant aux choix du fournisseur et des services PLM sont les manques que rencontrent les systèmes actuels et qui sont soulignés par les chercheurs. Des initiatives ont été menées pour pallier ces problèmes, et nous avons vu l'émergence de solutions orientées web. Dans ce cadre, nous avons présenté un nouveau paradigme : le *Cloud Manufacturing* (CM), tentant comme

le CC d'apporter une solution en terme de livraison de service manufacturier sur internet afin de permettre la gestion efficace de projets PLM. L'introduction de ces paradigme nous a permis d'annoncer notre problématique de recherche et les questions sous-jacentes ainsi que la démarche envisagée pour y répondre.

3

Le Cloud Manufacturing

Résumé du chapitre

Nous avons présenté dans le précédent chapitre la notion du PLM et avons expliqué que l'état de l'art révélait que les systèmes actuels ne sont pas assez avancés pour assurer une flexibilité et un passage à l'échelle suffisants pour suivre la volatitlié de la demande. Dans ce chapitre, nous présentons en détails le Cloud Manufacturing (CM), qui est un nouveau paradigme où toutes les ressources et les capacités manufacturières impliquées dans la gestion du cycle de vie du produit sont fournies à l'utilisateur sous forme de services via Internet. Trouvant ses racines dans des technologies telles que l'architecture orientée service (SOA) et le Cloud Computing (CC), le CM est une solution émergente où les utilisateurs peuvent demander des services allant de la conception de produits, la fabrication, etc. Nous présentons par la suite les architectures qui ont été proposées dans la littérature pour le CM et comparons ensuite les travaux qui traitent des systèmes manufacturiers orientés Web afin de vérifier s'il existe une architecture CM dans les travaux scientifiques.

Sommaire

3.1 Vers de nouveaux modèles manufacturiers : Grid Manufacturing

3.1.1 Définition

Nous avons présenté dans le chapitre 1 le *Cloud Computing* (CC) en tant que nouveau paradigme offrant un accès à l'utilisateur, via Internet, à un ensemble de ressources dont l'utilisation peut être ajustée à échelle réduite ou agrandie selon le besoin. Nous avons expliqué en quoi cela peut être bénéfique pour les utilisateurs surtout les petites et moyennes entreprises qui manquent d'infrastructure tant il leur permet une réduction considérable des coûts car il paient à la demande ce qu'ils ont réellement consommé. [Fan 04] explique que dans l'industrie, la conception et la fabrication de produit devient de plus en plus complexe que les entreprises devraient utiliser toutes les ressources qui se trouvent en interne ou en externe pour satisfaire son objectif et sa stratégie. La vision des modèles en réseau telles que les entreprises en réseau a donné lieu à un nouveau paradigme : les *grilles de production* [1] (de l'anglais *Manufacturing Grid (MGrid)*). Le terme Grid a été inventé au milieu des années 90 pour désigner une infrastructure de calcul distribuée au service des sciences de l'ingénieur [Foster 99] car elle permet le partage et l'agrégation d'une grande variété de ressources informatiques distribuées mais liées via Internet [Baker 02]. Les grilles ont été utilisées dans l'informatique et l'astronomie pour offrir une puissance de calcul aux utilisateurs comme dans le projet SETI@home [2] ou pour la gestion et le partage de mégadonnées (big data) comme dans le projet Globus [3].

Dans la même veine que les grilles de calcul, le MGrid a été proposé comme un modèle permettant le partage dynamique de ressources entre les entreprises en utilisant Internet. Les MGrid utilisent les TIC pour dépasser les limites de la collaboration causées par la distance géographique, rendant ainsi les différentes ressources entièrement connectées [Liu 08, Tao 07].

Selon [Fan 04] la MGrid peut être défini comme :

Définition 3.1 *la MGrid est un environnement integré supportant à la fois le partage et l'integration de ressources manufacturières dans l'entreprise pour la gestion et l'activité collaboratives de l'entreprise. [Fan 04].*

Par conséquent, nous pouvons conclure à partir de la définition ci-dessus que les MGrid est un réseau virtuel manufacturier qui offre des capacités manufacturières comme les réseaux électriques fournissent de l'énergie. Ils visent à résoudre le problème du partage de ressources distribuées et hétérogènes afin d'entreprendre et d'exécuter des opérations dans un environnement coopératif [Liu 08]

1. Nous utilisons le terme *Manufacturing Grid* dans ce travail au lieu de *Grille de production*
2. http://setiathome.berkeley.edu consulté en Octobre 2015
3. https://www.globus.org consulté en Octobre 2015

3.1.2 Architecture et fonctionnement des MGrid

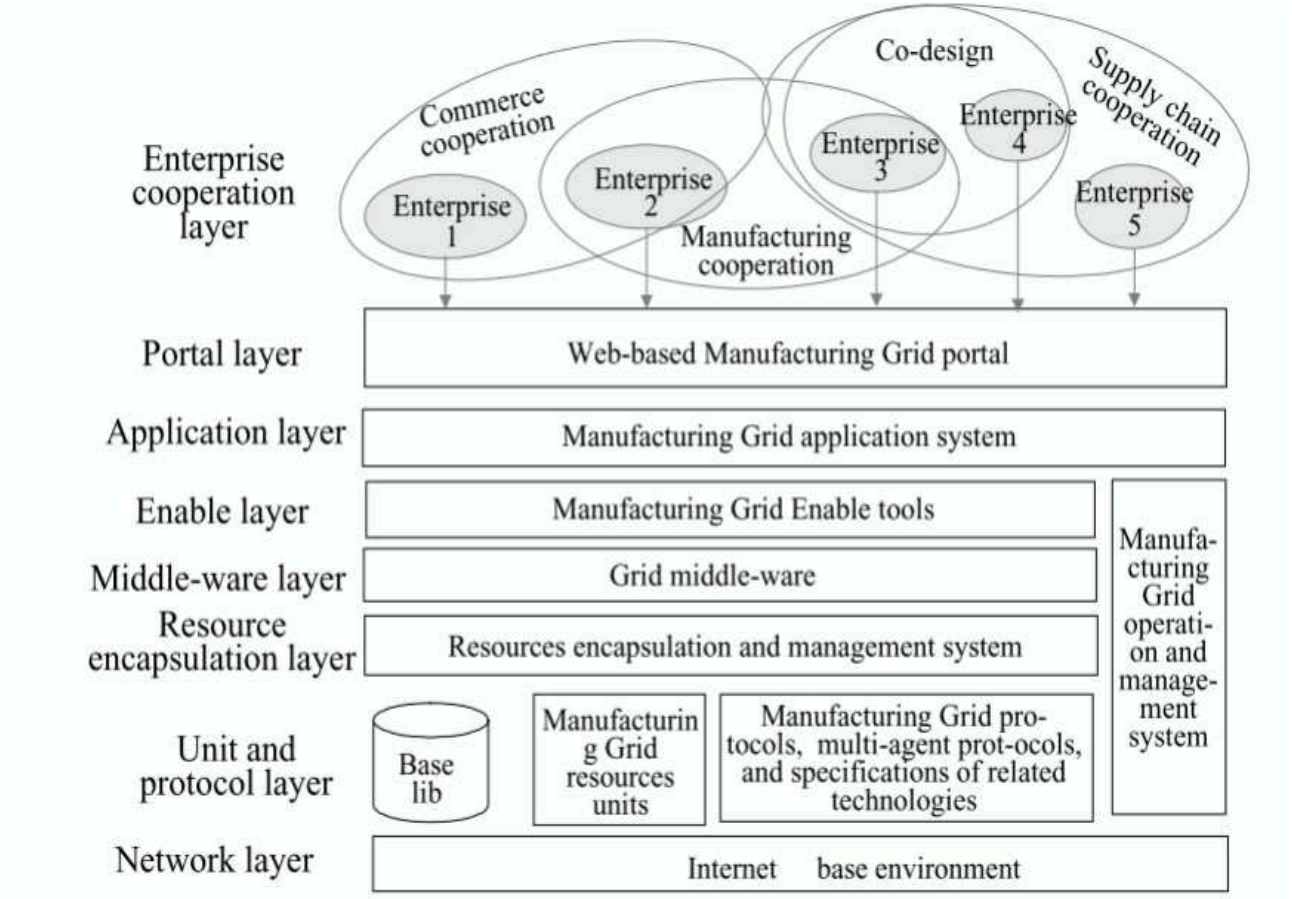


FIGURE 3.1 – *L'architecture d'une MGrid selon [Fan 04]*

Un système MGrid possède une architecture en couches comme le montre la figure 3.1. Les fonctions de chaque couche sont décrites comme suit [Fan 03] :

1. **Couche réseau** : fournit l'environnement de communication de base à toutes les ressources et les organisations dans le systèmes MGrid.
2. **Couche protocle** : sert de support pour la construction et l'exploitation de la MGrid.
3. **Couche d'encapsulation de ressource** : encapsule les ressources en utilisant des technologies spécifiques qui permettent de convertir les ressources locales en ressources globales gérées par le système de gestion des ressources.
4. **Couche middleware** : fournit les fonctions de bases de la MGrid comme l'allocation et l'accès aux ressources.
5. **Couche d'activation** : contient des outils permettant le developpement et la gestion de la MGrid, comme les outils de gestion des enregistrements, l'ordonnancement des ressources, etc.
6. **Couche application** : contient les applications manufacturières développées pour la

grille en question, comme par exemple des *Product Data Management* (PDM)[4], applications de gestion commerciale, etc.

7. **Couche portail** : offre une interface via laquelle l'utilisateur peut se connecter à la MGrid.
8. **Couche de coopération des entreprisess** : représente les projets de collaboration propres à chaque entreprise.

[Tao 10] explique que dans une MGrid, les tâches (ou requêtes pour des services manufacturiers) peuvent être classées en deux types : (a) demande de service pour une seule ressource, (b) demande de service multi-ressources. Généralement, une tâche de type (a) est exécutée en invoquant une seule ressource tandis qu'une tâche de type (b) s'exécute en invoquant plusieurs ressources. Les auteurs poursuivent ; dans le cas d'une demande pour ressource de type (a), le système cherche la ressource qui permet de répondre à la requête de façon optimale, alors que pour une requête de type (b) la tâche est décomposée en plusieurs sous-tâches de type (a). Ainsi, pour le second type de requête, le système doit trouver le chemin optimal entre les différentes ressources et services qui répond aux critères de l'utilisateur comme par exemple maximiser la fiabilité, minimiser les coûts, etc. Ce dernier fonctionnement correspond à un problème connu sous le nom de *Multi-Objectives MGrid Resource Service Composition and Optimal-Selection* (MO-MRSCOS) qui consiste à trouver la composition de services optimale dans le but de résoudre une fonction multi-objective [Tao 08].

3.1.3 Les caractéristiques d'un réseau manufacturier

[Fan 04] explique que les services requis par les réseaux manufacturiers sont différents des services d'information fournis par Internet, il dresse ainsi une liste caractérisant les besoins des réseaux manufacturiers :

— **interaction :** le service manufacturier doit assurer l'interaction entre les utilisateurs et les services ;
— **temps réel :** le service manufacturier dois refléter l'état actuel des équipements et répond aux requêtes des utilisateurs en temps réel ;
— **Multi-coopération :** l'utilisateur doit être en mesure d'intégrer différent services manufacturiers. Les demandes de celui-ci doivent être satisfaites par le réseau formé par les ressources ;
— **Long cycle de vie :** en comparaison avec les services d'information, certains services manufacturiers peuvent avoir un long cycle de vie ;
— **Volume de données :** les services manufacturiers ont besoin de transférer une grande quantité de données comparés aux services d'information qui gèrent généralement des pages web ;
— **Structure et fonction complexes :** la structure du système fournissant les services manufacturiers est relativement complexe ;
— **Spécialisation et connaissance :** les services manufacturiers doivent avoir un grand degré de spécialisation et basé sur une connaissance spécifique quant au domaine traité.

4. Un système de gestion de données techniques, ou SGDT, est un ensemble d'outils informatiques pour la gestion des données techniques liées à un projet de conception.
Ces outils ont pour objectifs de remplir les fonctions suivantes : stocker, gérer et contrôler toutes les informations et processus concernant la définition, la production et la maintenance d'un produit. L'acronyme correspondant en anglais est PDM, pour Product Data Management.

3.2 Le *Cloud Manufacturing* (CM)

À partir du CC, un nouveau modèle orienté "Manufacturing" a été proposé : Le Cloud Manufacturing [1] (CM) [Xu 12b, Li 10a, Tao 11]. Xu [Xu 12b] explique que le CM est une solution attrayante car le modèle manufacturier n'est plus orienté produit mais plutôt service. Nous introduisons brièvement le CC et ses principales caractéristiques dans la Section suivante.

3.2.1 Le Cloud Manufacturing : Définition

Nous avons présenté précédemment le CC et avons expliqué qu'il permet de réduire les coûts des infrastructures IT ce qui est bénéfique pour les petites et moyennes entreprises. En effet, contrairement aux grandes entreprises, les petites/moyennes entreprises ont besoin de techniques plus rentable pour gérer leurs activités et l'infrastructure sous-jacente en raison de leur budget plus réduit et de leur peu de connaissance dans les IT. De plus, elles sont constamment en croissance, et leurs systèmes d'information peuvent évoluer selon leur passage à léchelle, objectifs, modes de production et de gestion [Xu 08]. Selon Ming et al. [Ming 07], les nouvelles technologies doivent être développées pour permettre aux entreprises de livrer des produits adaptés aux besoins des clients avec un coût de production faible. Ouzrout [Ouzrout 12] estime que la prochaine génération de modèles et d'architectures doivent tenir compte des nouveaux modèles informatiques tels que les solutions à la demande et le CC. Le même auteur confirme que l'externalisation et l'utilisation des applications "en ligne" favorise l'agilité des entreprises, facilite l'échange et le partage de données, et permet de créer de nouveaux services de collaboration à faible coût.

[Fuchs 08] expliquent l'avènement de l'Internet a conduit à la création de réseaux collaboratifs, impliquant un changement dans le modèle d'organisation hiérarchique. Ces modèles, selon les auteurs, ne peuvent plus soutenir l'innovation : « *À une époque où la collaboration de masse peut remodeler l'industrie, les vieilles habitudes hiérarchiques de l'organisation du travail et de l'innovation ne se prêtent plus au niveau de l'agilité, de la créativité, et la connectivité dont les entreprises ont besoin pour demeurer concurrentielles.* » Ainsi, l'industrie a besoin d'un modèle, soutenu par la mondialisation, permettant aux organisations d'être connectés

Site	Nombre d'article
Sciencedirect	1
Springer	12
Google Scholar	90

TABLE 3.1 – *Résultats de la recherche des articles en utilisant l'expression "Cloud Manufacturing" - Fin 2012*

Le CM est un nouveau concept qui vise à apporter les avantages du CC à l'industrie manufacturière. Le tableau 3.1 présente le nombre de travaux liés au CM, issus de littérature scientifique qui ont été recensé durant l'année 2012. Il est ainsi important de souligner le nombre réduits d'articles traitant de ce domaine sachant que dans google Scholar il y a les doublons des articles trouvés sur les autres sites en plus des brevets et d'autres citations. En se basant sur la définition du [Mell 09] du CC, de nombreux auteurs ont proposé des définitions au CM. Parmi eux, nous citons : [Li 10a], [Zhang 14b], [Zhang 10a], [Xu 12b], [Wu 12], et

1. Le terme "Manufacturing" représente le domaine manufacturier de manière générale avec toutes les étapes du cycle de vie et non seulement l'étape de production.

[Schaefer 12]. Le terme *Cloud Manufacturing* (CM) a été utilisé d'abord par [Li 10a] en 2010. [Wu 13a] propose une définition du CM qui, tout comme [Xu 12b], se base sur la définition du NIST [Mell 09] pour proposer une définition au CM qui englobe aussi les avantages de celui-ci :

Définition 3.2 *Le CM est un modèle centré sur l'utilisateur qui offre un accès à la demande à un ensemble de ressources manufacturières diversifiées et distribuées pour former des lignes de productions temporaires et reconfigurables, et qui permet d'améliorer l'efficacité, réduire les coûts du cycle de vie et de répondre à la demande variable des utilisateurs [Wu 13a].*

3.3 Vision stratégique du CM

[Tao 11] explique que, dans le CM, en plus de fournir des ressources informatiques (IT), l'objectif est de mettre à la disposition de l'utilisateur des ressources manufacturières sous forme de différents services : design as a service (Daas), manufacturing as a service (MFGaas) experimentation as a service (Eaas), simulation as a service (SIMaaS), management as a service (Maas), maintain as a service (MAaas) et integration as a service (INTaas). La relation entre CC et CM est illustrée dans la figure 3.2. Chacune de ces différentes formes de service fait appel aux trois niveaux de services du CC : Saas, Paas et Iaas.

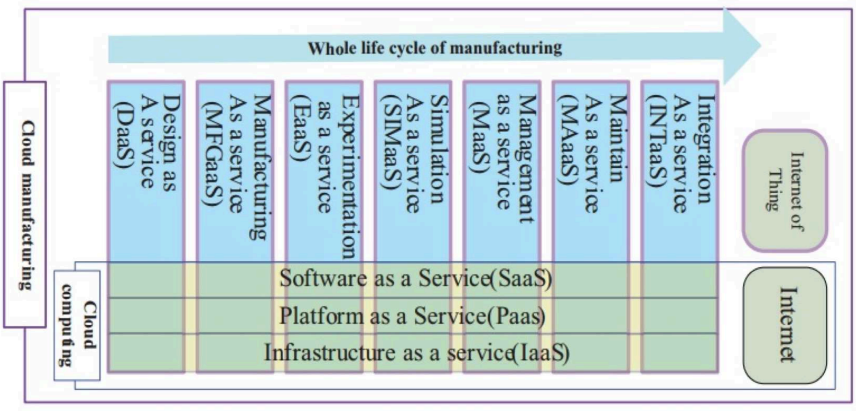


Figure 3.2 – *Relation entre CC et CM [Tao 11]*

La figure 3.3 montre qu'un utilisateur peut demander un logiciel de conception (Design as a Service) et une infrastructure de stockage (Iaas) afin de stocker ses données.

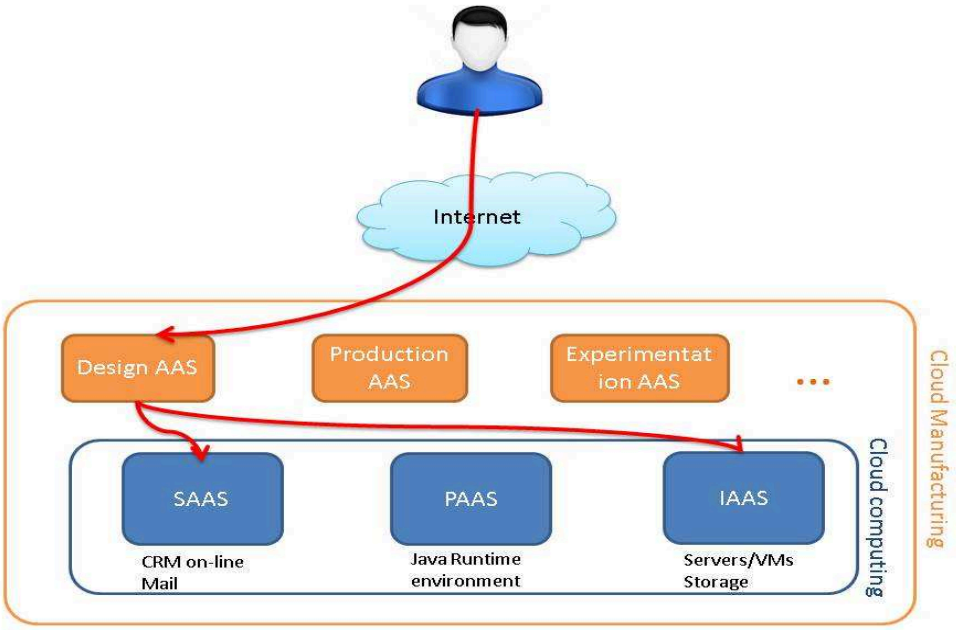


FIGURE 3.3 – *Exemple d'utilisation du CM*

[Wu 13a] expliquent que *le CM possède un modèle d'interaction fournisseur-client (Figure 3.4). Ainsi, il nécessite une interaction entre trois entités : les utilisateurs (consommateurs), les fournisseurs d'applications et les fournisseurs de ressources physiques. Les besoins de l'utilisateurs seront mappés avec les capacités offertes par les ressources via la couche application. Ce modèle représente l'offre-demande du marché qui motivera l'existence du CM.*

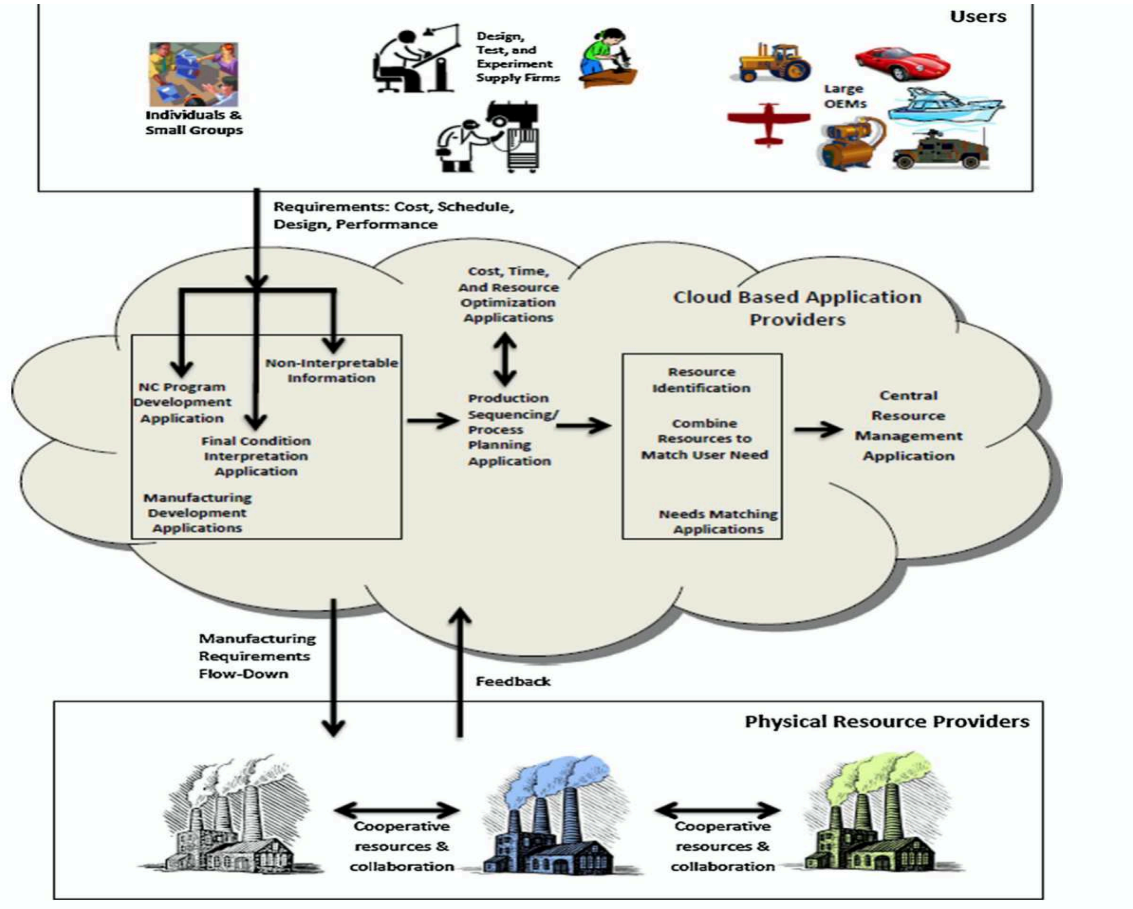


FIGURE 3.4 – *Vision stratégique du CM selon [Tao 11]*

Les auteurs exposent les caractéristiques clés pour promouvoir le concept du CM et sa mise en place [Wu 13a] :

— Solution centrée sur le client : tandis que le fonctionnement des chaines de provisionnement dans le 21^{me} siècle est un processus hiérarchique, le CM offre une solution flexible à l'utilisateur qui aura la possibilité de choisir entre plusieurs fournisseurs celui ou ceux qui répondent à son besoin. Ainsi, les contrats de sous-traitance deviennent rigides car les utilisateurs sont sévèrement limités par le choix des fournisseurs et sous-traitants. D'autre part, lorsque les relations entre ces parties n'est plus souhaitable, il s'avère très souvent difficile et couteux de les dissoudre. Le CM dans ce cas offre au consommateur d'améliorer l'expérience de manière à réduire les coûts et améliorer la qualité.

— Ressources temporaires, reconfigurables et dynamiques : une autre caractéristique distinctive du CM est la nature dynamique et flexible de l'approvisionnement des ressources. Par conséquent, les lignes de production du CM sont destinées à être de nature temporaire, permettant la production de petits lots sans exclure les longs cycles de production. La capacité de reconfigurer rapidement et réutiliser les ressources manufacturières permet une haute efficacité, réduit au minimum les temps d'arrêt et offre une réponse dans les meilleurs délais à la demande. De ce fait, la flexibilité du système reposera sur cette capacité là. Pour accomplir une telle tâche, un niveau élevé d'automatisation sera nécessaire pour veiller à ce que la répartition des tâches se fait sur les ateliers adéquat avec un minimum d'effort. L'automatisation ne signifie pas l'absence

d'être humain dans le processus car ce dernier pourrait interagir avec le système pour assurer la qualité et la prévention des erreurs.
— Possibilité de réaliser tous types de tâches : grâce à une large gamme de ressources physiques, des tâches qui étaient autrefois pas viable économiquement peuvent être d'actualité grâce à la flexibilité de l'environnement. Les applications basées sur le CM permettent d'établir de multiples scénarios de coûts et d'ordonnancement pour le consommateur en utilisant l'accès à un ensemble de ressources de manière à rendre réalisables des tâches qui sont difficiles à accomplir dans un environnement traditionnel isolé.
— Système dirigé par la demande et la demande intelligente : comme toute entité manufacturière aujourd'hui, le CM est dirigé par la demande de l'utilisateur. Contrairement entreprises manufacturières traditionnelles, l'environnement CM sera basé sur la demande intelligente grâce à la flexibilité inhérente du système qui sera utilisé pour assurer un partage de charge sur les ressources de fabrication équivalents ou interchangeables.

3.4 Architecture d'un système CM

Certains travaux, proposent des architectures de CM ; nous citons par exemple le projet ManuCloud, financé par le septième framework-program pour la commission européenne pour la recherche (FP7) qui selon [Meier 10] est un projet qui vise a permettre la création de réseaux manufacturiers intégrés couvrant plusieurs entreprises, en utilisant les technologies orientées services. Selon les mêmes auteurs, l'architecture CM, illustrée par la figure 3.5 offre aux utilisateurs la possibilité d'utiliser les capacités des réseaux manufacturiers configurables et virtualisés basés sur la fédération de plusieurs usines et soutenus par un ensemble d'application proposées en tant que services.

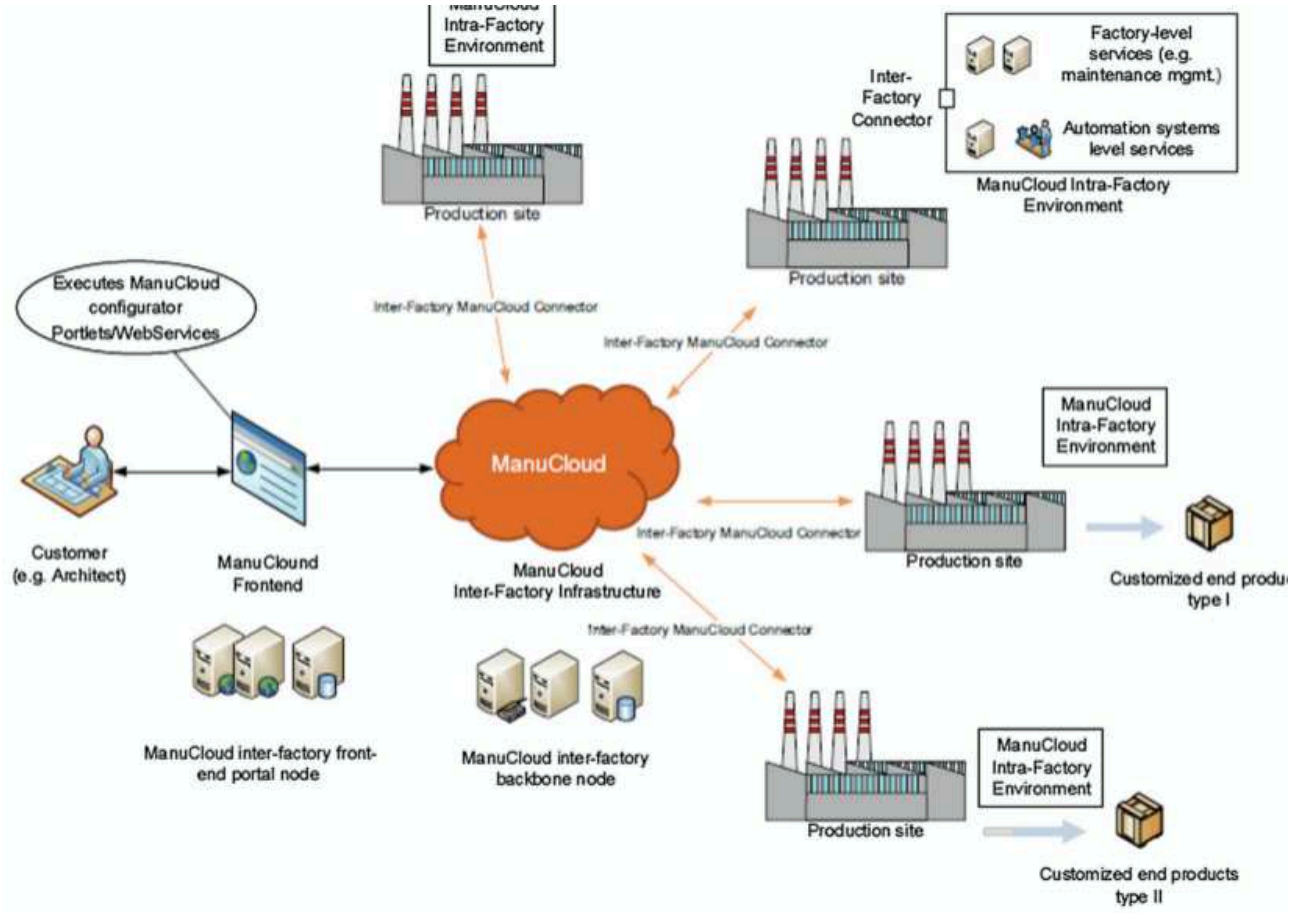


FIGURE 3.5 – *L'architecture ManuCloud selon [Meier 10]*

Dans la même veine, [Lv 12] a proposé une architecture CM (Figure 3.6) qui s'articule autour de quatre couches : couche physique, couche de connexion, couche virtuelle et la couche application. L'auteur explique que la couche virtuelle est la couche la plus importante dans l'architecture car elle permet l'intégration et le partage des ressources manufacturières. Dans cette couche se fait le mapping entre les ressources physiques et les ressources logiques, ainsi que la définition de l'interface standard permettant l'accès et le partage de la ressource physique. L'auteur poursuit, le médiateur du cloud et la couche de gestion ont la fonction de gérer les services comme l'inscription, la recherche et la réservation d'un service.

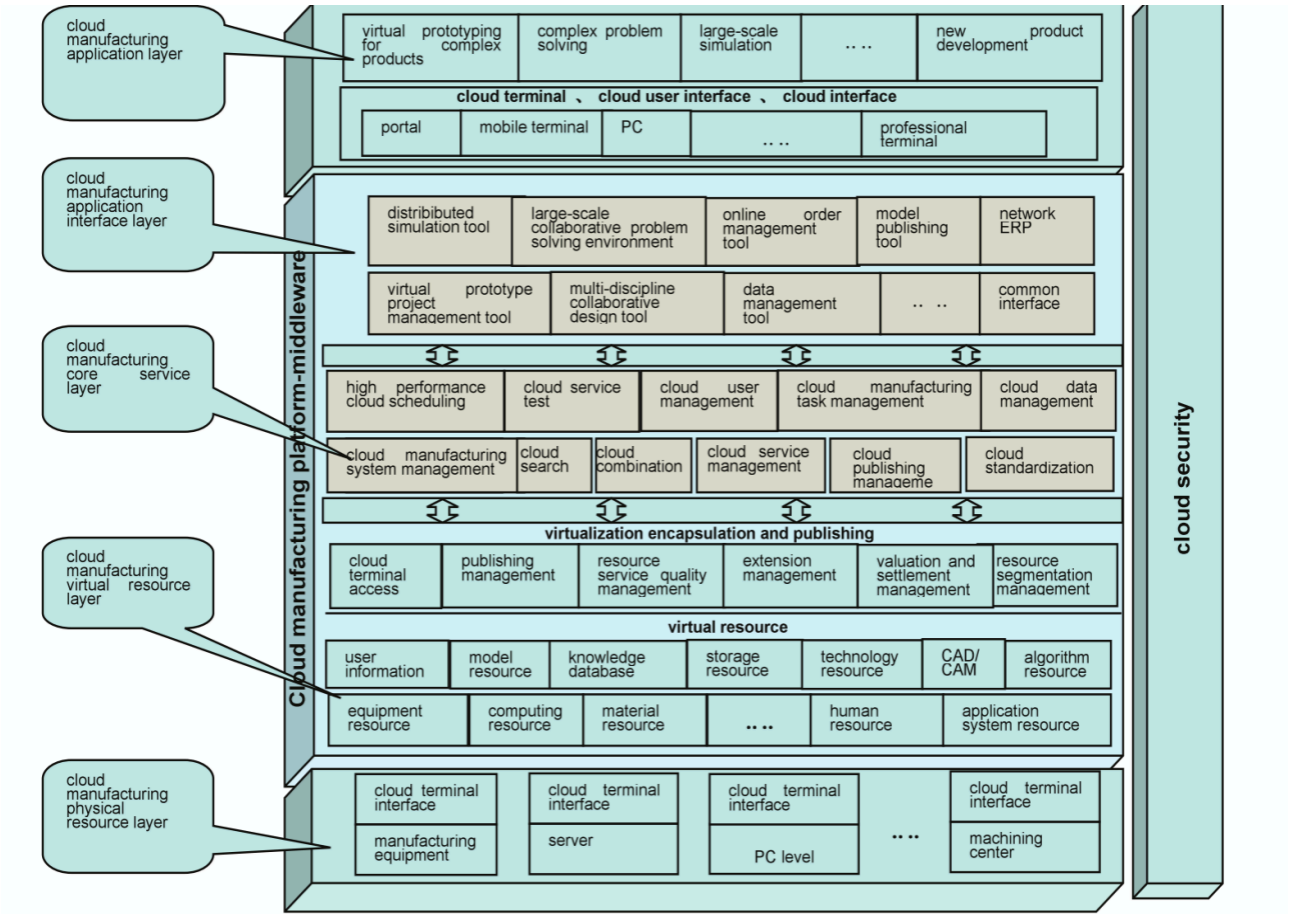


FIGURE 3.6 – *L'architecture ManuCloud selon [Lv 12]*

[Tao 11] présente une architecture composée de dix couches (Figure 3.7) :

1. couche de ressources : représente la ressource et la capacité manufacturières impliquées dans le cycle de vie ;
2. couche de perception : responsable de la détection des ressources physiques, ainsi elle leur permet d'être connectées au réseau manufacturier ;
3. couche de virtualisation : responsable de la virtualisation des ressources et de leur encapsulation dans des services ;
4. couche de service : fournit principalement deux catégories de service, le service manufacturier (manufacturing cloud service (MCS)) et le service de base du Cloud. Le MCS est le résultat de l'encapsulation des ressources et capacités manufacturières physiques tandis que le service de bases fournit les fonctionnalités qui peuvent être offertes par la plate-forme Cloud comme l'inscription, l'invocation de services, etc. ;
5. couche application : cette couche offre la possibilité de créer des applications CM qui suivent les besoins de l'entreprise comme un ERP basé sur les services CM ;
6. couche portail : fournit diverses interfaces d'interaction homme-machine pour les utilisateurs ;
7. coopération d'entreprises : représente les différentes applications qui peuvent être réalisées en utilisant les service du CM comme la conception collaborative ;

8. couche de connaissance : offre divers connaissances nécessaires pour les autres services, comme les connaissances liées au domaine manufacturier, modèles de connaissances, etc. ;
9. couche sécurité : fournit des mécanismes, stratégies et méthodes pour le système CM ;
10. couche Internet élargie : fournit l'environnement de communication de base pour toutes les ressources, utilisateurs et les services.

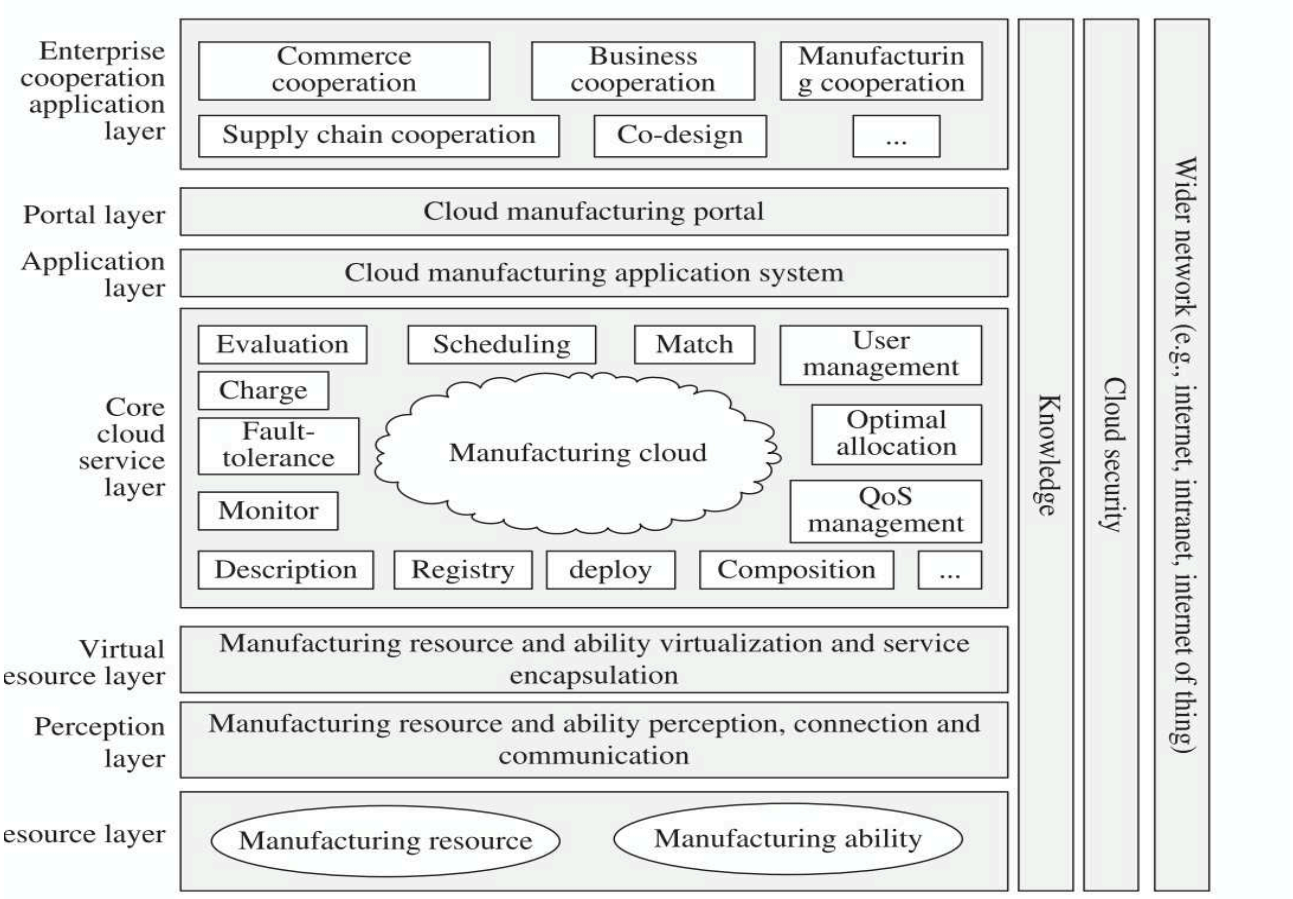


FIGURE 3.7 – *L'architecture ManuCloud selon [Tao 11]*

Inspirée par les travaux de [Tao 11], les auteurs dans [Zhang 14a] proposent une version plus compacte de celle-ci, qui reprend les mêmes couches présente dans l'architecture proposée par [Tao 11]. Cette architecture est représentée par la figure 3.8.

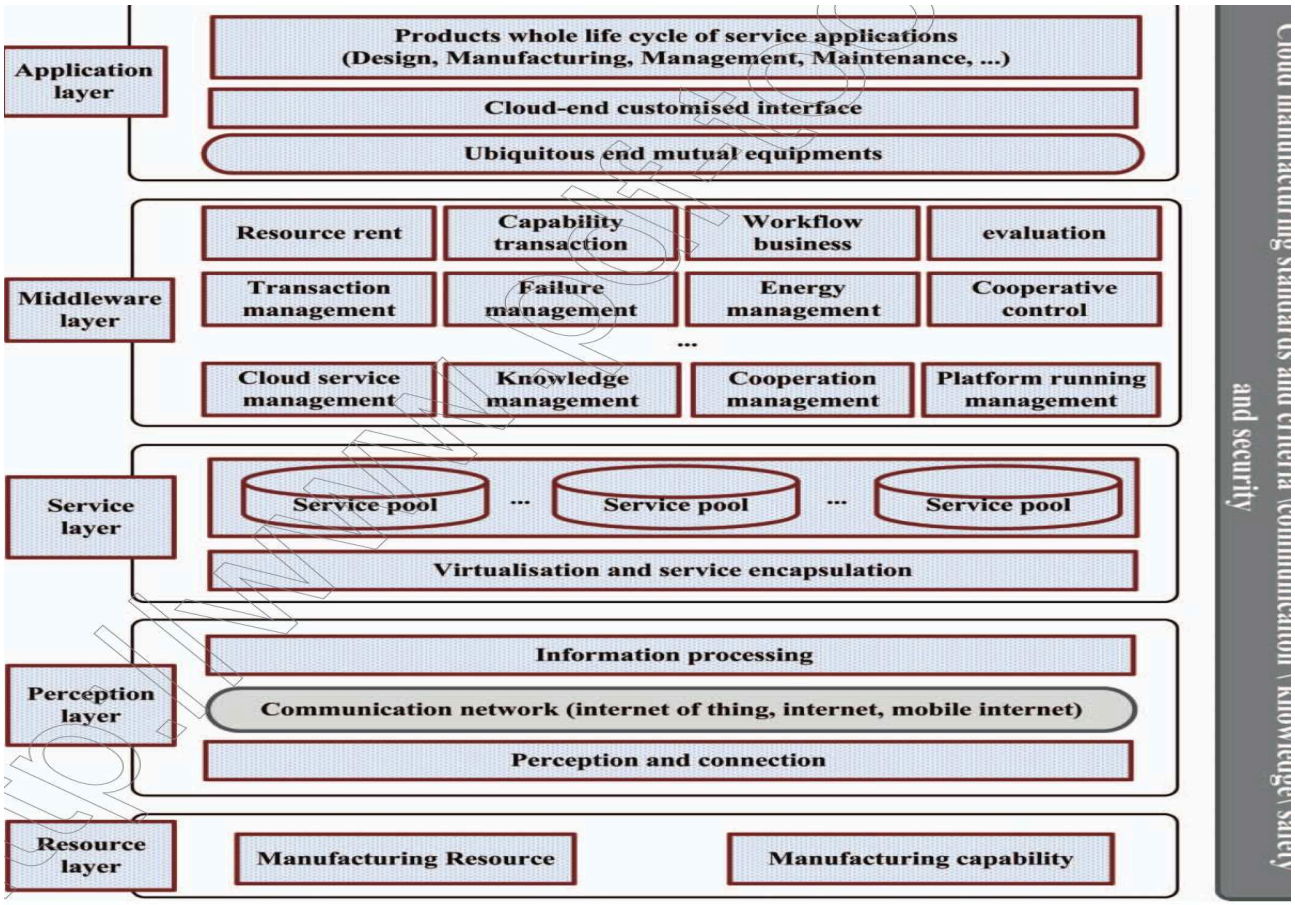


FIGURE 3.8 – *L'architecture ManuCloud selon [Zhang 14a]*

L'architecture de référence dans la littérature scientifique[5] est celle proposée par [Xu 12b] dans son article « *From Cloud Computing to Cloud Manufacturing* » où il expose les conditions requises pour mettre en place un système CM. Ainsi un système CM se compose de quatre couches (Figure 3.9) [Xu 12b] :

5. Cette affirmation se base sur le nombre de citations de l'article de [Xu 12b] qui compte 446 citations sur le site de google scholar consulté en Septembre 2015.

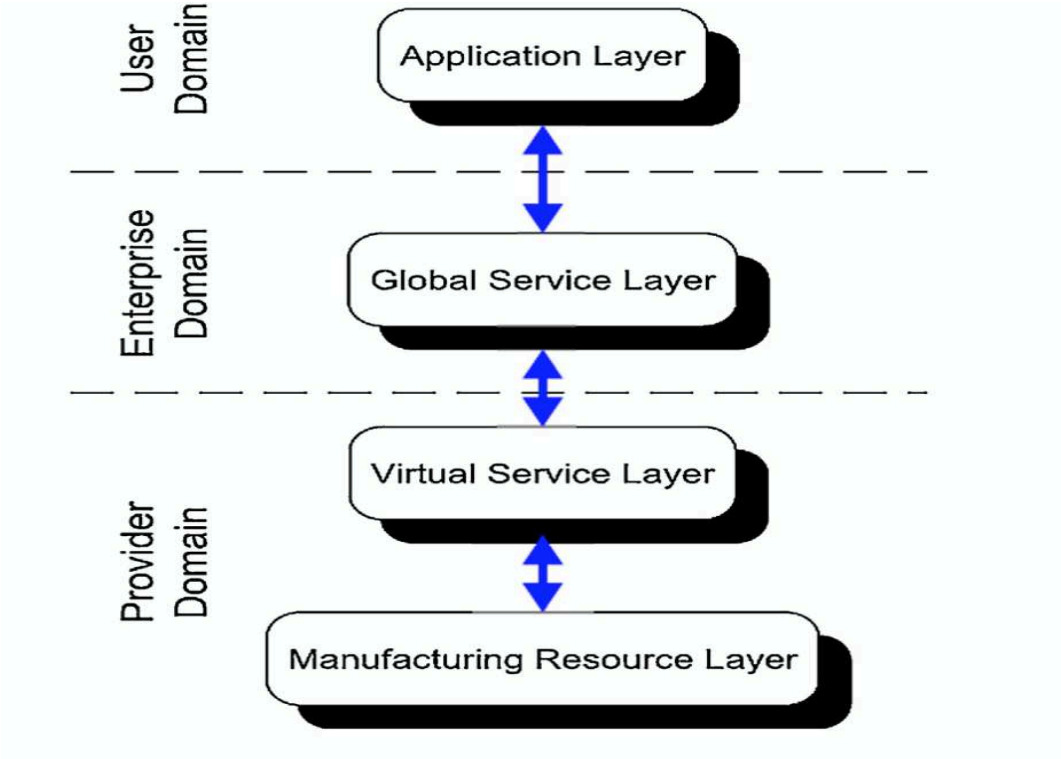


FIGURE 3.9 – *Cadre architectural du Cloud Manufacturing [Xu 12b]*

1. la couche des ressources (*Manufacturing Resource Layer* (MRL)) : contient les ressources nécessaires au cours du cycle de vie. Ces ressources manufacturières peuvent être sous deux formes : ressources physiques et capacités. Les ressources physiques peuvent être de type matérielles : comme les serveurs, les équipements, ordinateurs, matière première, etc. ou logicielles comme les logiciels et outils de simulation mais aussi tout le savoir faire et les ressources humaines. Les capacités manufacturières sont intangibles et dynamiques et représentent les compétences d'une organisation. Celles-ci peuvent inclure des capacités de conception, de simulation, d'expérimentation, de production, de gestion et d'entretien. A ce niveau le type de services fournis peut être Iaas et Saas ;
2. la couche virtuelle (*Virtual Service Layer* (VSL)) : où les ressources manufacturières identifiées sont virtualisées et empaquetées pour fournir un service. Les fonctions clés à ce niveau sont : (a) identification des ressources, (b) virtualisation des ressources identifiées et (c) les empaqueter en tant que service CM. L'auteur ([Xu 12b]) explique qu'il est beaucoup plus difficile de réaliser ses fonctions sur une application CM que pour une application CC. Des technologies comme la *Radio Frequency IDentification* (RFID) [Brahim-Djelloul 12, Huang 07], les réseaux de capteurs sans fils [Huang 09], Internet des objets, les sytèmes cyber-physique [Trentesaux 15] peuvent être utilisées pour identifier les ressources. Comme nous l'avons précédemment définie (section 1.3.3), la virtualisation se réfère à l'abstraction des ressources logiques des ressources physiqyes sous-jacentes. Les ressources manufacturières sont généralement mappées à des machines virtuelles indépendantes du systèmes, gérées par le gestionnaire de virtualisation responsable de la communication avec la couche physique et de l'allocation des machines virtuelles. [Xu 12b] explique que les systèmes multi-agents ont le potentiel de fournir environnement plug-and-play et facilite ainsi l'intégration des ressources virtalisées. [Michaloski 09, Vijayaraghavan 08, Vijayaraghavan 09] est un exemple d'outil per-

mettant ceci. L'étape suivante consiste à empaqueter la ressource virtualisée pour créer un service CM. Ceci est réalisable en utilisant des langages et protocoles de description comme le STEP resource locator (STRL) [Campos 09, Xu 09] qui permet, dans un réseau manufacturier utilisant la norme *STandard for the Exchange of Product model data* (STEP) [Paviot 11, Pratt 05], de représenter un service CM. Il y a trois méthode de virtualisation [Xu 12b] :

— One-to-one : s'applique à une ressource manufacturière qui ne fournit qu'une seule fonction et donc est directement virtualisée en un seul service ;
— Many-to-one : dans ce cas, plusieurs ressources (où chacune fournissant une fonction spécifique) peuvent être combinées pour créer un service améliorée. A ce stade, la question de trouver la composition optimale de service doit être soulevée. [Zhang 10b, Guo 10] traitent de cette question et discutent les problèmes de composition de ressources et la flexibilité de ce processus ;
— One-to-many : concerne une ressource qui apparaitra comme un ensemble de ressources virtualisées. C'est le cas d'une ressource partagée entre plusieurs utilisateurs.

3. la couche globale (*Global Service Layer* (GSL)) : c'est une série de technologies permettant le développement d'une plate-forme cloud. L'Internet des objets par exemple l'interconnexion entre les périphériques et le produit. Cela dit, l'auteur préconise la mise en place d'un régime ou système de gestion centralisée et efficace pour fournir aux entreprises manufacturières des services agiles et dynamiques.

Définition 3.3 *L'agilité signifie le fait d'utiliser la connaissance du marché et l'entreprise virtuelle pour exploiter les opportunités rentables dans un marché volatile [Lemieux 13, Ben Naylor 99].*

Définition 3.4 *L'agilité est une capacité de l'entreprise à l'échelle qui englobe les structures organisationnelles, systèmes d'information, des processus logistiques et, en particulier, les mentalités. Une caractéristique clé d'une organisation agile est la flexibilité [Christopher 00].*

Dans la couche GSL deux modes de services peuvent êtres mises en place : (a) un mode complet où le GSL est responsable de toutes les opérations et activités du cloud comme la virtualisation et le monitoring des ressources, ou (b) un mode partiel, où le fournisseur de service gère des activités additionnelles comme la gestion des équipements manufacturiers. En général, le GSL est responsable de la location, l'allocation et le monitoring des ressources manufacturières. Les fournisseurs d'équipements manufacturiers sont toujours responsables de l'exécution des tâches et de la qualité de celle-ci.

4. la couche application (*User Domain* (UD)) : sert d'interface entre l'utilisateur et les ressources du Cloud Manufacturing. Elle fournit au client des terminaux et interfaces pour utiliser la plate-fore CM. Comme pour le CC, l'utilisation des services doit être mesurable et facturable. Activity-Based Costing [Louth 09] est un outil qui peut être utilisé par le client pour comprendre et estimer le coût de l'utilisation des service CM. Le problème à ce niveau, reste encore une fois comme pour le CC celui de la confidentialité et le stockage des données personnelles et/ou sensibles des utilisateurs et entreprises. Il existe des technologies qui permettent de résoudre de tels problèmes comme [Rimal 10] la compression et le cryptage de données stockées et l'utilisation des VPN pour assurer un accès à distance sécurisé. [Xu 12b] explique qu'un SLA rigoureusement établi est une condition nécessaire pour gagner la confiance de l'utilisateur final surtout dans le cas

du CM ou une défaillance de matériel de production par exemple est généralement plus difficile à récupérer.

3.4.1 Différence entre MGrid et CM

La différence entre le CM et la MGrid est que cette dernière est étroitement liée aux réseaux manufacturiers qui sont une agrégation de ressources pour effectuer une tâche spécifique. En effet, un réseau manufacturier se réfère à « un ensemble de ressources intégrées prêtes à être utilisées » tandis que le CM reflète l'idée que « les ressources intégrées sont distribuées et prête à être utilisées » [Cheng 10]. Cela signifie que, dans CM, plusieurs ressources sont proposées comme un ensemble de services qui sont prêts à gérer des tâches différentes. Dans la même veine, [Xu 12b] explique que ce qui manque dans les systèmes manufacturiers en réseau aujourd'hui est la gestion centralisée des services et le choix de différents modes de fonctionnement. Les principales différences entre CM et MGrid sont représentés dans le tableau 3.2.

Caractéristique	MGrid	CM
On-demand	No	Yes
Network access	Yes	Yes
Resource Pooling	Yes	Yes
Rapid elasticity	No	Yes
Measured service	Yes	Yes
Different tasks	No	Yes

TABLE 3.2 – *La différence entre la MGrid et le CM*

Dans la section suivante, nous présentons un ensemble de travaux qui ont été recensés durant l'état de l'art et qui représentent des solutions basées sur le web, liées au CM pour les comparer à l'architecture définie par [Xu 12b]. La collecte de ces travaux à eu lieu durant l'année 2013 et témoigne encore une fois du nombre réduit d'articles traitant de ce domaine .

3.5 Systèmes Cloud Manufacturing : État de l'art

Dans cette Section, nous comparons différents travaux qui proposent des solutions liées au CM et qui sont résumées dans le tableau 3.3. Notre objectif est de savoir si elles remplissent toutes les exigences du cadre architectural proposé par [Xu 12b] (figure 3.9). Certains travaux proposent des plate-formes modulaires et configurables pour gérer l'hétérogénéité entre plusieurs logiciels [Brecher 09, Van Der Velde 09, Wang 12]. Ces solutions sont basées sur SOA qui garantit l'intégration des différents modules représentant les services offrant ainsi une modularité et une réutilisation au système. Avec la même perspective, [Mokhtar 10] ont étudié une plate-forme manufacturière distribuée et ont prposé une méthodologie pour générer une "raodmap" entre logiciels de conception. D'autre travaux proposent des solutions multi-agents orientées service dont le but est de permettre le management et le contrôle des agents manufacturiers distribués [Nagorny 12] ou d'assurer l'intégration de données de conception dans un environnement distribué en utilisant STEP [Valilai 13]. STEP permet de décrire d'une

manière non ambigue des données du produit pendant toutes les phases du cycle de vie, indépendamment de tout système informatique.

Afin d'optimiser l'utilisation des ressources partagées, des méthodes sont proposées dans la littérature pour assurer : une allocation optimale des ressources informatiques en CM[Laili 12], le partage des ressources dans un environnement de CM [Wu 10] et une composition optimale de services basé sur la QoS et le coût [Sun 11] ; la QoS fournit une garantie de performance, de disponibilité, de sécurité et de fiabilité.

TABLE 3.3 – *CM travaux connexes [Talhi 13]*

Paper	UD	GSL	VSL	MRL	Proposal
[Brecher 09]	X	X	X		Module-based platform for interoperable CAD-CAM-NC chain
[Van Der Velde 09]	X	X			Plug-and-Play framework for construction of simulation software by selecting target simulation components
[Mokhtar 10]		X	X		Platform for interoperability between CAx
[Suh 08]	X	X			a concept for management of product design and manufacturing information during the lifecycle
[Wang 12]	X		X	X	Service-oriented platform to enable integration of softwares and package them as a service
[Valilai 13]	X	X	X		Service-oriented approach to achieve an integrated and collaborative platform for distributed manufacturing agents
[Nagorny 12]		X	X		Approach for implementing a service-and multi-agent-oriented architecture to manage distributed manufacturing componentes
[Laili 12]		X	X	X	Method for optimal allocation of computing resources in CM
[Wu 10]		X	X	X	Method for resource sharing in CM environment
[Sun 11]		X		X	Multi-objective method for an optimal service combination under Qos and cost criteria
[Xu 12a]	X	X		X	A framework that maps a user-appliation to a resource in CM taking the Qos in account
[Park 12]	X	X			Method to choose between software the Saas that meets user requirements

Bien que ces travaux proposent des solutions qui ont un lien avec le CM, ils ne remplissent pas toutes les conditions définies par [Xu 12b] afin de mettre en place une architecture robuste de prise en charge du CM (cf tableau 3.3).

[Xu 12b] distingue entre deux visions du CM :

— L'adoption de certaines technologies de CC dans les environnements manufacturiers ;

— le CM comme la version manufacturière et industrielle du CC.

Dans ces travaux, nous nous concentrons sur le deuxième point afin de proposer une méthodologie permettant la mise en place d'une plate-forme CM afin de mettre en correspondance les utilisateurs et les fournisseurs de ressources et solutions permettant la gestion du cycle de vie et facilitant la collaboration entre les acteurs du projet.

3.6 Conclusion

A la recherche de systèmes avancés permettant de suivre les demandes des clients de plus en plus volatiles, de nouveaux modèles manufacturiers tels que les MGrid ont vu le jour. Dans ce chapitre nous avons introduit ce modèle, en présentant l'architecture générale d'une MGrid ainsi que les caractéristiques d'un réseau manufacturier et les attentes que celui-ci devrait remplir. A partir de ce type de réseaux manufacturiers et du CC, une nouveau modèle orienté « manufacturing » a été proposé dans la littérature, il s'agit du CM. C'est un modèle ou les ressources manufacturières (logiciels et équipement) sont proposées à l'utilisateur sous forme de service, à la demande offrant ainsi une composition temporaire et dynamique de chaines de production qui suivent la demande du client. Nous avons exposé deux définitions du CM qui revenait souvent dans les travaux de recherches, ainsi que la vision stratégique de ce dernier, et les opportunités qu'il offre aux entreprises. Quelques architectures du CM ont été présentée suivies par l'architecture de référence proposée par [Xu 12b]. Nous avons comparé un ensemble de travaux liés au CM afin de vérifier s'il existe une implémentation du CM. Tous les travaux recensés s'insère dans différents niveaux de l'architecture de [Xu 12b] sans remplir toutes les conditions. Ainsi, notre objectif est de proposer une méthodologie permettant de mettre en place une telle architecture. Dans le chapitre suivant nous exposons notre choix de méthodologie permettant la réalisation de ce système et servant de support et de guide aux utilisateurs dans leur processus de migration vers le CM.

4

Méthodologies de modélisation et présentation de la méthodologie ASCI

Résumé du chapitre

Nous présentons dans ce chapitre un état de l'art sur les méthodologies et modélisation suivi par nos critères de selection de la méthodologie que nous souhaitons utilisée. Il s'agit de la démarche Analyse, Spécification, Conception, Implémentation (ASCI) sur laquelle nous souhaitons nous appuyer afin de proposer une méthodologie permettant la mise en place d'une plate-forme CM. Nous exposons ensuite les étapes parincipales sur lesquelles repose ASCI :

- *la décomposition systémique d'un système en trois sous-systèmes et la construction d'un modèle générique de connaissance du domaine étudié ;*
- *la construction d'un modèle de connaissance qui décrit la structure et le fonctionnement du système étudié ;*
- *l'obtention d'un modèle d'action qui est la traduction du modèle de connaissance dans un formalisme mathématique ou informatique ;*
- *la définition d'un modèle de résultats regroupant les critères nécessaires à la prise de décision.*

Nous passons en revue ensuite les travaux de litterature où les auteurs proposent des adaptations de ASCI dans différents domaines allant des systèmes du trafic urbain, les systèmes hospitaliers, les systèmes de production et le Green CC.

Sommaire

4.1 Méthodologies de modélisation : état de l'art

Une méthodologie de modélisation est définie comme étant :

Définition 4.1 *Une méthodologie de modélisation est un ensemble de méthodes, d'outils, d'approches et de concepts permettant de modéliser un système [Chauvet 09].*

Ainsi, une méthodologie de modélisation permet de donner un guide général à suivre sans donner précisément les outils qui devront être utilisés. Dans cette section nous effectuons un état de l'art des méthodologies de modélisation des systèmes, afin de trouver celle qui correspond le mieux à notre objectif.

4.1.1 PERA

Purdue Enterprise Reference Architecture (PERA) [Williams 98] est une méthodologie d'ingénierie des environnements industriels développée à l'université de Purdue aux USA. Elle offre la possibilité de modéliser les composants humains ainsi que les composants manufacturiers d'une entreprise, mais aussi de modéliser les informations et le contrôle du système. Selon [Williams 98] c'est une méthode nouvelle et unique pour définir la place de l'humain dans l'entreprise « informatisée ». La méthodologie définit cinq phases du cycle de vie d'une entité industrielle depuis sa conceptualisation jusqu'à sa mise en opération en passant par les phases de conception :

- conceptualisation,
- définition,
- conception,
- installation et de construction,
- opérationnelle et de maintenance.

La phase de conceptualisation peut se diviser en deux étapes :

- l'étape d'identification,
- l'étape de concepts.

4.1.2 La méthodologie ASCI

La démarche *Analyse, Spécification, Conception, Implémentation* (ASCI) (figure 4.1) a été proposée pour concevoir une méthodologie de modélisation et son environnement logiciel. Elle permet donc la conception et la mise en œuvre d'outils d'aide à la décision appelés modèles d'action (modèle de simulation, modèles métaheuristiques,...) [Gourgand 91]. Elle permet de concevoir la modélisation d'une classe de systèmes : le modèle générique de connaissance de cette classe, et de réaliser une bibliothèque de composants logiciels réutilisables qui est exploitée pour générer un modèle d'action (programme informatique) pour un système de la classe. Ainsi, le cadre conceptuel de la méthodologie de modélisation ASCI s'appuie sur un processus de modélisation séparant explicitement le recueil de la connaissance de son exploitation [Huet 11]. Pour construire le modèle générique de connaissance des systèmes étudiés, ASCI préconise une décomposition systémique en trois sous-systèmes [Chabrol 08] : *Sous-Système Physique (SSP), Sous-Système Logique (SSL)* et *Sous-Système Décisionnel (SSD)*.

La démarche ASCI se compose de quatre étapes (figure 4.1) : les étapes d'Analyse et de Spécification permettent d'obtenir le modèle générique de connaissance tandis que les phases

de Conception et d'Implémentation permettent la création de la bibliothèque de composants logiciels réutilisables.

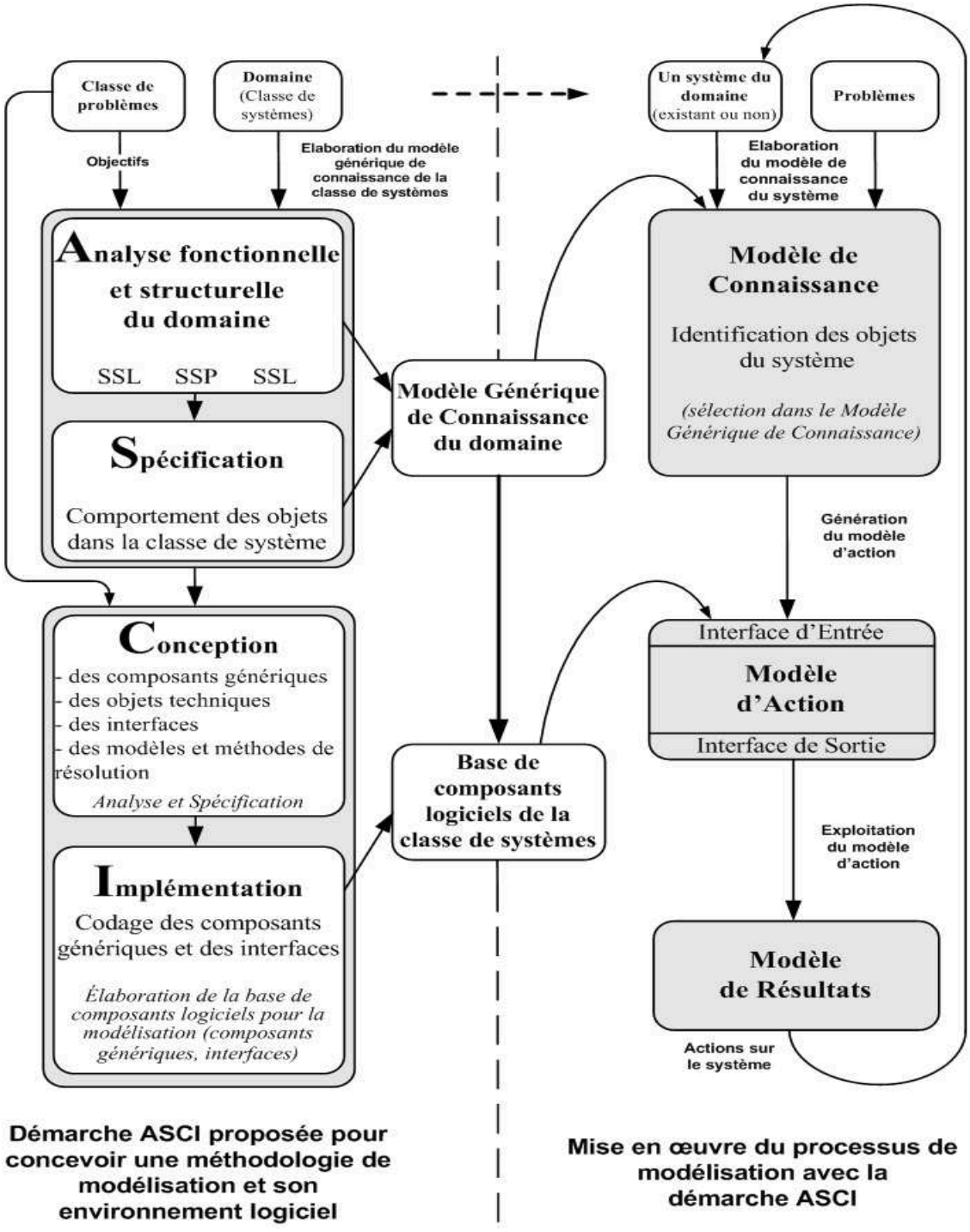


FIGURE 4.1 – *La méthodologie ASCI*

4.1.3 MBCSA

Le *Méthodologie Bidirectionnelle de Conception des Systèmes Automatisés* (MBCSA) a été proposé par [Chiron 08] et développée au sein du LIMOS afin de fournir une vision bidirectionnelle du processus d'analyse et de développement centrée sur la modélisation de composants multi-facettes avec le langage *Unified Modeling Language* (UML) [1] et l'extension *Systems Modeling Language* (SysML) [2], selon le cadre architectural *Model Driven Architecture* (MDA) [OMG 03]. Plus précisément, c'est une méthodologie basée sur (a) l'approche par composants, développée pour le génie logiciel pour permettre une réutilisation du code [Brown 96] et le (b) MDA. Selon [Servat 05], la MBCSA s'intéresse à la modélisation structurelle du composant en deux vues principales : une vue en boîte noire décrivant les services fournis et requis par ce composant, découlant de l'attribution des collaborations ; et une vue en boîte blanche qui décrit l'assemblage des composants internes le constituant. [Chiron 08] préconise la séparation des spécifications en deux mouvements distincts à savoir (Figure 4.2) :

— Une **Analyse Objet Montante (AOM)** ne se préoccupant pas du contexte de réalisation et se concentrant sur la spécification d'entités élémentaires. L'AOM passe par les phases d'identification, de spécification détaillée, de validation et de test, puis de capitalisation dans des bibliothèques.

— Une **Analyse Objet Desendante (AOD)** pour la réalisation de projet avec la réutilisation obligatoire d'éléments définis dans le premier cas. Les participants de l'AOD travaillant sur un projet donné, partent d'une analyse des besoins, puis choisissent les composants à réutiliser, les connectent dans une phase de conception, puis les déploient vers les plates-formes cibles.

1. UML est un langage à but universel, proposé par le consortium Object Management Group (OMG). UML ne propose pas de méthode mais seulement une liste de diagrammes composés en deux types : « structurel » et « comportemental ». Les diagrammes structurels montrent la structure statique des objets dans un système tandis que les diagrammes comportementaux montrent le comportement dynamique de ces objets incluant leurs méthodes, leurs collaborations, leurs activités et leurs historiques des états.

2. SysML est un profil d'UML2 qui étend son champ d'application initial, le génie informatique, à l'ingénierie des systèmes. Son objectif est de supporter la spécification, l'analyse, la conception, la vérification et la validation d'une grande quantité de systèmes complexes comme des systèmes physiques, des logiciels, des systèmes d'information, des systèmes automatisés, des processus, des personnes et des biens.

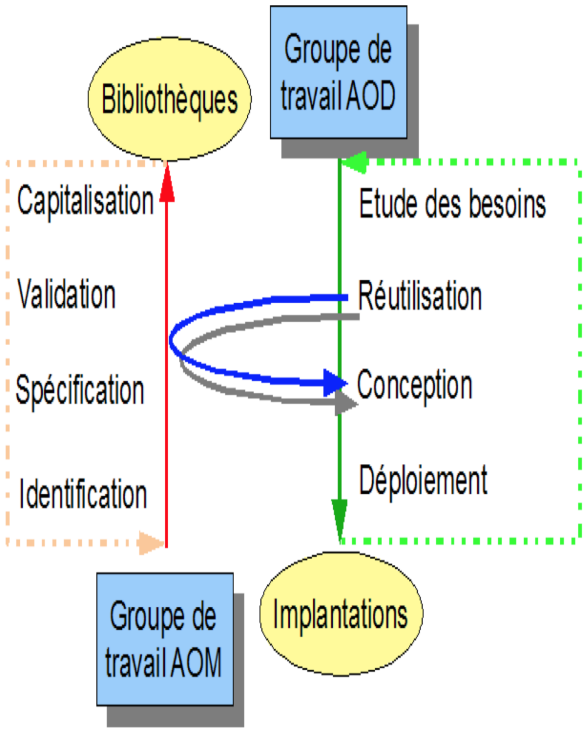


FIGURE 4.2 – *Processus d'analyse bidirectionnel [Chiron 08]*

4.1.4 TROPOS

TROPOS est une méthodologie de développement de logiciel orientés agents fondée sur deux caractéristiques clés : (a) les notions d'agent, de buts, de projets et d'autres concepts et connaissances qui sont des primitifs fondamentaux pour le développement des logiciels et (b) un rôle important assigné à l'analyse des exigences et spécifications des systèmes [Giunchiglia 03]. Elle est composée de cinq phases pour développer un système multi-agents [Bresciani 04, Giorgini 04] :

1. analyse préliminaire des besoins : permet de comprendre le problème en étudiant ses configurations organisationnelles, faire une analyse orientée-but et modéliser les intentions de chaque acteur comme buts. Les acteurs sont des entités sociales dépendantes les unes des autres pour exécuter des tâches, fournir des services et atteindre des objectifs ;
2. analyse avancée des besoins : dans cette phase il est question de décrire le système dans son environnement opérationnel, construire un modèle stratégique rationnel décrivant comment réaliser les buts identifiés dans l'étape précédente. Ce modèle est un graphe qui contient quatre types de nœuds : objectif, tâche, ressource et objectif affiné, ainsi que deux types de liens : décomposition et méthode ;
3. conception architecturale : définir l'architecture globale du système en termes de sous-systèmes inter-connectés via les données, le contrôle et les dépendances ;
4. conception détaillée : sert à définir le comportement de chaque composant et les communications entre les agents ;
5. implémentation : fait le lien entre les spécifications de la plate-forme d'implémentation et les notions développées dans la phase précédente de conception détaillée.

L'architecture TROPOS est composée de quatre niveaux [Bresciani 04] :

— Méta-méta-modèle : représente les éléments structurels du langage.
— Méta-modèle : qui est une instance du méta-méta-modèle et permet de définir les notions du niveau de connaissance.
— Domaine : une instance du méta-modèle qui représente l'application à une entité d'un domaine.
— Instance : est une instanciation des éléments du domaine.

[Huet 11] explique que l'inconvénient de TROPS réside dans le fait d'être une méthodologie de développement de logiciel orientée agent. Ainsi, elle nécessiterait de nombreuse adaptations pour pouvoir être appliquée à un système manufacturier dont les entités possèdent une partie matérielle (comme les holons par exemple).

4.1.5 GAIA

GAIA [Wooldridge 00] est une méthodologie où le système multi-agent est vu comme une organisation composée de rôles interagissant entre eux. C'est la première méthodologie complète proposée pour l'analyse et la conception des systèmes multi-agents. La méthodologie prend comme point de départ une description de la structure organisationnelle du système, composée de rôles et d'interaction entre eux. Deux phases sont distinguées (Figure 4.3) : la phase d'analyse qui permet d'identifier les rôles du système étudié et de modéliser leurs interactions, et la phase de conception durant laquelle sont créés le modèle d'agents spécifiant les types d'agents, le modèle de service spécifiant les services à implémenter par les types d'agents et le modèle d'association décrivant les liens de communication entre les agents. Cette méthodologie aborde la conception d'un système multi-agent sans un modèle d'architecture d'agent préétabli mais celle-ci est très faible puisque aucun modèle computationnel [3] n'est envisagé[Nfaoui 08]. Selon [Pavón 06] il n'y a pas non plus de guides méthodologiques pour assigner les rôles aux agents. Plusieurs extensions de cette méthodologie ont été proposées pour surmonter les limitations de la version d'origine comme par exemple [Cernuzzi 06] qui propose des solutions pour améliorer l'adaptabilité de cette méthodologie tout au long du cycle de vie du système multi-agent.

3. Un modèle computationnel est un modèle logique ou mathématique permettant de simuler des activités cognitives selon le site http://memovocab.perso.sfr.fr consulté en Septembre 2015

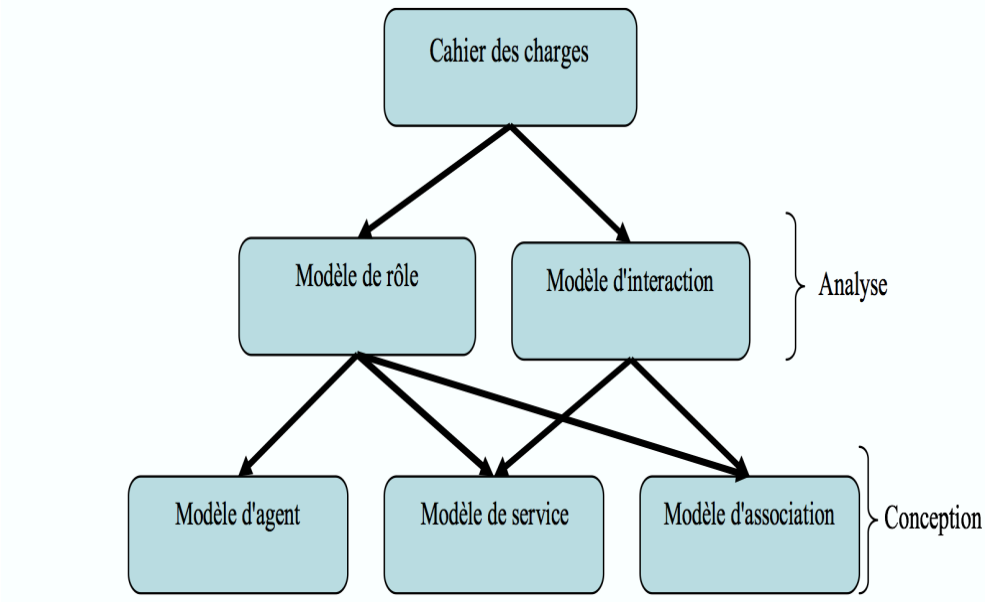


FIGURE 4.3 – *La méthodologie GAIA [Ali 09]*

4.2 CRITÈRES DE SELECTION

Notre but étant de définir une méthodologie qui permet de mettre en place une plateforme CM il est par conséquent nécessaire d'avoir un cadre méthodologique expliquant les étapes à suivre pour atteindre cet objectif. Selon [Zellner 11], une méthodologie doit comporter :

1. Un modèle procédural : une séquence d'activités à réaliser.
2. Un ensemble de techniques : comment générer les livrables en support des activités.
3. Des résultats : un artefact (par exemple un document, etc.) créé par une activité.
4. Des rôles : l'identification des acteurs et de leurs responsabilités.
5. Un modèle informationnel de la méthodologie : un modèle qui décrit les éléments de la méthodologie et leurs relations.

D'autre part, [Bernus 97] expose les exigences que toute architecture ou méthodologie de référence d'entreprise doivent satisfaire :

— Un meilleur traitement du domaine de l'entreprise d'un point de vue théorique : il est nécessaire que toutes les activités qui sont impliquées, directement ou indirectement, dans la conception et l'amélioration de l'entreprise doivent être prises en compte par l'architecture. Étant donné que l'objectif est de fournir une méthodologie (pas seulement une architecture) l'architecture doit être l'épine dorsale de la méthodologie. Ainsi l'architecture doit être fondée sur la modélisation des processus de l'entreprise et de son cycle de vie ;

— La mise à disposition d'un environnement de modélisation cohérente conduisant à un code exécutable : les vues de modélisation fournies doivent couvrir un ensemble minimal mais qui peut être extensible avec de nouvelles vues connexes. Ces vues de modélisation doivent être fondées sur une théorie commune ou un méta-modèle à travers lequel les vues peuvent être liées. L'environnement de modélisation idéal devrait être modulaire de sorte que les méthodologies alternatives pourraient être basées sur ce

dernier. Aussi, l'environnement de modélisation doit être extensible plutot que d'être un ensemble fermé de modèles.

— L'existence d'une méthodologie détaillé que l'entreprise peut suivre : en plus d'être techniquement correcte, la méthodologie doit être compréhensible et utilisable par la communauté ciblée. Ainsi, la méthodologie doit être exécutable par les équipes, garantissant un résultat de haute qualité avec des contraintes de coûts, délais et ressources acceptables. Une méthodologie d'ingénierie d'entreprise devrait également être extensible, puisque de nouvelles méthodes peuvent être intégrées, il est par conséquent nécessaire que le cadre méthodologique puisse anticiper cela. Il devrait aussi être possible de présenter la méthodologie à la fois de manière générique mais aussi dans ses formes détaillées et spécialisées. Ces formes spécialisées pourraient mieux servir dans des domaines particuliers de l'industrie par exemple.

— L'adoption de bonnes pratiques d'ingénierie pour le développement de modèles réutilisables, testés, et standard.
Il est important que la complexité apparente du processus d'ingénierie de l'entreprise doit être maintenue faible. Les détails complexes de modèles devraient être encapsulés dans des blocs de construction réutilisables. L'intégration et l'ingénierie de l'entreprise sont des processus complexes réalisés par un groupe de personnes. En tant que tel, la méthodologie devrait idéalement être basé sur une théorie permettant la conception collaborative entre plusieurs agents. Cette base théorique vise à assurer la pérennité de la méthodologie.

— Fournir une perspective d'unification, pour les processus, la gestion, les produits, le développement de l'entreprise et la gestion stratégique : le développement de l'entreprise doit être considéré comme l'une des activités de l'entreprise. Ainsi, l'architecture doit relier l'intégration et l'ingénierie de l'entreprise avec les restes des activités de l'entreprise.
L'architecture générique de référence de l'entreprise et la méthodologie doivent définir clairement la façon dont d'autres efforts pour l'intégration leur sont liés (par exemple : la réingénierie des processus, le travail collaboratif par ordinateur, la gestion de la qualité totale, entreprise virtuelle, les flux de travail, etc.).
Il est permis, voire souhaitable, d'avoir plusieurs présentations de l'architecture pour faciliter la compréhension et l'acceptation par une variété de groupes d'utilisateurs et de développeurs.

La liste ci-dessus peut être complétée par des exigences fonctionnelles, organisationnelles ou physiques/techniques.

En nous basant sur la définition de [Zellner 11] et les exigences ci-dessus, nous avons identifié la méthodologie ASCI comme adéquate à notre démarche pour répondre à notre problématique de recherche. Celle-ci est présentée dans la section suivante.

4.3 Présentation d'ASCI

4.3.1 Définition

La démarche *Analyse, Spécification, Conception, Implémentation* (ASCI) [Gourgand 91] a été développé au sein du LIMOS et est issue des travaux de modélisation sur les systèmes complexes (système de trafic urbain, système de production, systèmes hospitaliers, ...). ASCI (figure 4.1) a été proposée pour concevoir une méthodologie de modélisation d'un domaine et son environnement logiciel. Elle permet donc la conception et la mise en œuvre d'outils

d'aide à la décision appelés modèles d'action (modèle de simulation, modèles métaheuristiques,...) [Gourgand 91]. Elle permet de concevoir la modélisation d'une classe de systèmes : le modèle générique de connaissance de cette classe, et de réaliser une bibliothèque de composants logiciels réutilisables qui est exploitée pour générer un modèle d'action (programme informatique) pour un système de la classe.

Définition 4.2 *Un domaine est une classe de systèmes qui regroupe un ensemble de systèmes comparables ayant les mêmes caractéristiques techniques et fonctionnelles. Il est ainsi possible de décrire la classe des systèmes industriels, la classe des systèmes hospitaliers, etc. [Rodier 10].*

La démarche ASCI s'articule autour de deux axes majeurs :
— les quatre étapes pour modéliser un domaine : Analyse, Spécification, Conception et Implémentation ;
— le processus de modélisation pour un système du domaine.

4.3.2 Les différentes étapes de la méthodologie ASCI : Modélisation du domaine

4.3.2.1 Le modèle générique de connaissance

Le modèle générique de connaissance pour un domaine défini est développé durant les phases successives :

— **L'analyse fonctionnelle et structurelle du domaine** qui formalise le domaine sous une forme graphique ou syntaxique et consiste à rechercher les entités du domaine, leurs fonctions et leurs associations [Rodier 10]. Pour construire ce modèle, ASCI préconise une décomposition systémique qui vient compléter celle proposée par [Le Moigne 92] en trois sous-systèmes communicants (Figure 4.4) :

  i Sous-Système Physique (SSP) : représente l'ensemble des ressources physiques (qui pourraient concerner différents domaines tels que : la production, le stockage, les transports, ...), leur répartition géographique et les relations qui les lient.

  ii Sous-Système Logique (SSL) : représente le flux (transactions) entre les entités que le système doit traiter, ainsi que l'ensemble des opérations relatives à ces flux.

  iii Sous-Système Décisionnel (SSD) : qui est structuré en centre de gestion. Il contient les règles de gestion et du fonctionnement du système.

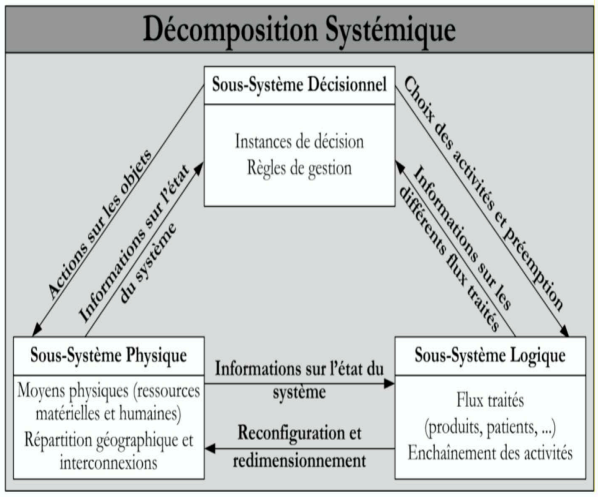


FIGURE 4.4 – *Décomposition du système en trois sous-système [Rodier 10]*

— **La spécification des entités et de leurs comportements** durant cette phase il est question d'exprimer le fonctionnement des entités du domaine et des flux. Cette phase permet de décrire le fonctionnement du système. La spécification ainsi réalisée doit permettre aux experts en modélisation et aux experts du système de s'accorder sur le fonctionnement (réel ou désiré) du domaine [Rodier 10].

4.3.2.2 La bibliothèque de composants logiciels

La création d'une bibliothèque de composants informatique à partir du modèle générique de connaissance se fait en deux étapes préliminaires :

— **La conception** des composants logiciels génériques consiste à formaliser un ensemble de composants sans prendre en compte les détails de l'implémentation comme les langages par exemple . Ces composants seront utilisés pour construire un modèle d'action.

— **L'implémentation** il s'agit de développer dans un langage choisi les composants définis durant la phase de conception. Les composants codés prennent la forme de modules logiciels, réutilisés dans de différents modèles d'action et réalisés dans différents langages de programmation ou à partir de logiciels de simulation.

4.3.3 Les différentes étapes de la méthodologie ASCI : Modélisation d'un système du domaine

4.3.3.1 Le modèle de connaissance

Est un modèle qui décrit la structure et le fonctionnement d'un système du domaine étudié. Cette description peut être dans un langage graphique ou naturel et est fondé sur la connaissance décrite dans le système générique de connaissance. Pour un système existant (étude a posteriori), le modèle de connaissance contient la connaissance acquise après observations du système. Pour un modèle à concevoir (étude a priori), le modèle de connaissance contient les spécifications du futur système.

4.3.3.2 Le modèle d'action

Le modèle d'action est une traduction du modèle de connaissance dans un formalisme mathématique ou dans un langage de programmation permettant l'évaluation des critères de performances choisis. Plusieurs modèles d'action peuvent être construits à partir du même modèle de connaissance. La construction et l'utilisation consécutives de ces deux modèles constituent le processus de modélisation (Figure 7.11).

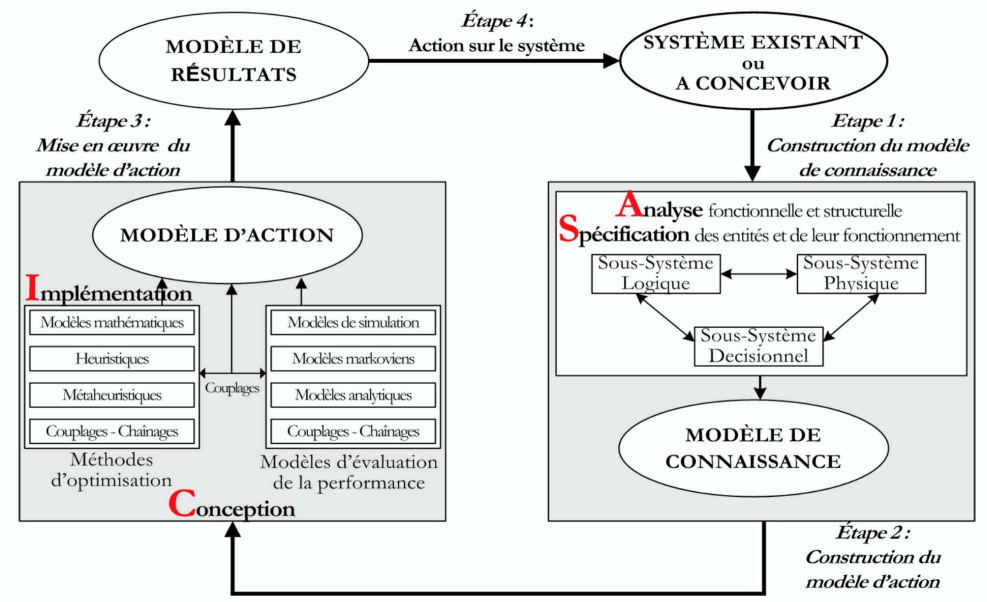


FIGURE 4.5 – *Le processus de modélisation [Rodier 10]*

4.3.3.3 Le modèle de résultats

Ce modèle est défini en amont, au moment du choix des critères de performance et de préférence avec les utilisateurs afin de répondre aux objectifs fixé. Il est alimenté à partir du modèle d'action et contient les résultats quantitatifs et qualitatifs fournis par celui-ci comme par exemple des indicateurs de performance. En fonction des résultats obtenus, les décideurs vont pouvoir évaluer la pertinence des solutions envisagées et choisir ou non de modifier la structure de leur système (décomposition systémique), leurs processus (comportements des entités) ou leurs règles de gestion [Rodier 10].

4.4 État de l'art

Récemment, plusieurs auteurs proposent des adaptations d'ASCI appliquées à des domaines différents :

— [Féniès 06] considèrent les systèmes de santé comme une chaîne logistique (Health care Supply Chain). Ils définissent la HSC comme un ensemble ouvert croisé avec des flux humains, matériels, financiers et d'information. Ils proposent une méthodologie de modélisation pour l'évaluation de la chaîne logistique nommée : ASCI-HSC.

— [Chabrol 08] ont développé une méthodologie pour concevoir des outils d'aide à la décision basée sur des différentes méthodes de résolution (simulation, formulation mathématique,...), qui commencent toutes par la description d'un modèle de connaissance

du système étudié. Ils présentent les différentes étapes à suivre depuis la modélisation jusqu'à la création d'outils d'aide à la décision, ainsi qu'une application de la méthodologie au sein des services de chirurgie et de gynécologie-obstétrique.

— ASCI a été adapté pour la modélisation du nouvel hôpital d'Estaing à Clermont-Ferrand [Rodier 10]

— [Mebrek 08] a utilisé ASCI afin de proposer une méthodologie de modélisation pour la conception de modèles de simulation de deux organisations hospitalières. Ces modèles permettent de : modéliser l'ancienne et la nouvelle structure de l'hôpital, simuler les deux organisations, étudier le dimensionnement des ressources matérielles et humaines, évaluer le taux d'utilisation des ressources, et évaluer le temps d'attente et la durée d'exécution des tâches.

— [Chabrol 06] proposent d'utiliser ASCI pour le trafic Multi-agent (ASCI-mi). Leur objectif est la mise en place d'un environnement logiciel d'aide à la décision pour les systèmes de trafic urbain. Leur méthodologie a été développée pour combiner plusieurs modèles (afin d'obtenir ce qu'on appelle un modèle hybrid). Les modèles peuvent représenter différents niveaux de détails ainsi que différents horizons temporels. Ils utilisent principalement le langage UML.

— [Ali 09] propose une méthodologie ASCImi-STU SMA (ASCI multiple, incrémentielle pour les système du trafic Urbain)pour le développement d'un modèle multi-agent pour le pilotage des STU. A la suite de cette adapation d'ASCI, un outil d'aide à la décision facilitant l'interactivité homme-machine, permettant de fournir à l'utilisateur des informations et des suggestions à été développé.

— [Galland 03] proposent une approche méthodologique pour la création de modèles de simulation de systèmes industriels pour les systèmes complexes et distribués. Cette approche, appelée MA MA-S facilite la modélisation et la simulation des processus de prise de décision, que celle-ci soit centralisée ou distribuée.

— [El Haouzi 08a] expliquent que les exigences relatives au contrôle des systèmes de production évoluent des approches traditionnelles centralisées où la prise de décision est hiérarchique aux architectures distribuées et complexes impliquant des entités et processus autonomes. Les auteurs ont proposé une approche méthodologique, basée sur ASCI, pour développer une bibliothèque de composants génériques de simulation qui peuvent être instanciés de manière automatique dans un modèle de simulation modulaire. Ils ont proposé de rajouter un diagramme décrivant les processus autonomes dans la décomposition systémique. Ils utilisent aussi le langage UML.

— [Huet 13b] proposent une méthodologie inspiré de l'approche par composants, le paradigme de système de systèmes et la méthodologie ASCI. Cette méthodologie permet de construire un modèle d'action avec l'objectif d'évaluer la performance d'un système manufacturier holonique et de gérer la prise de décision dans un environnement centralisé et décentralisé. Les diagrammes UML sont utilisés pour la modélisation.

— [Huet 13a] utilise ASCI dans le domaine du Green Cloud Computing. Les solutions actuelles essaient de prendre en compte de nombreux phénomènes selon un certain niveau de planification pour la gestion énergétique dans les centres de données (stratégique, tactique ou opérationnel). Pour faire face à cette complexité, les auteurs proposent une méthodologie de modélisation visant à construire un environnement logiciel. Le langage UML est utilisé pour la modélisation des systèmes de gestion et de contrôle.

— [Royer 14] propose d'utiliser un autre langage de modélisation : Business Process Model and Notation (BPMN). Leur objectif est de construire un outil d'aide à la décision

pour aider les pharmaciens à réorganiser le processus d'utilisation des médicaments dans un hôpital.

A partir de ces travaux, nous constatons que la méthodologie ASCI possède deux avantages : elle permet de créer une bibliothèque de composants réutilisables et l'évaluation d'un scénario sur un système du domaine ainsi que la constitution d'un guide pour le développement d'outils d'aide à la décision.

4.5 Conclusion

Nous avons passé en revue dans ce chapitre quelques travaux sur les méthodologies de modélisation. Notre objectif étant de proposer une plateforme CM qui permettra la mise en correspondance des fournisseurs et des clients dans le cadre d'une collaboration autour des projets PLM, nous avons expliqué le choix de la méthodologie que nous jugeons adéquate pour réaliser notre objectif. Nous avons ensuite présenté la méthodologie choisie : ASCI en expliquant les étapes par lesquelles doit passer la modélisation d'une classe de système afin de fournir un modèle générique de connaissance et une bibliothèque de composants logiciels et la modélisation d'un système de cette classe en utilisant les deux modèles précédemment définis. Cette section a été suivie par un état de l'art mettant en exergue la diversité des domaines où ASCI a été utilisé : système du trafic urbain, systèmes hospitaliers, contrôles des systèmes de production et le domaine du Green CC.

Nous proposons dans le chapitre suivant une adaptation de la méthodologie ASCI afin de développer un modèle générique représentant le domaine CM.

5

Proposition d'instanciation de la démarche ASCI au domaine du *Cloud Manufacturing* (CM) fondée sur les ontologies

Résumé du chapitre

Ce chapitre présente notre contribution méthodologique. Dans un premier temps, nous présentons et expliquons nos choix d'outils de modélisation. Nous utilisons les ontologies pour représenter notre modèle générique de connaissance. Les ontologies sont un outil de modélisation issu du Web sémantique. Ce choix est motivé par les raisons suivantes :

- *les ontologies proposent une vue simplifiée et compréhensible du domaine ;*
- *les ontologies permettent d'automatiser la découverte de service en effectuant des mappages sémantiques ;*
- *les ontologies parmettent la gestion des données non-canoniques.*

ASCI est adaptée pour concevoir un modèle générique de connaissance du domaine étudiée : le CM. Le langage OWL DL est utilisé pour représenter le modèle générique de connaissance basé sur les ontologies : CMO. OWL DL représente un compromis en terme d'expressivité et de raisonnement entre les trois langages d'ontologie : OWL-Full, OWL DL et OWL lite. Nous présentons ensuite les étapes de constructions de notre CMO et les concepts et propriétés qui la composent. Nous finissons ce chapitre par l'exposition d'un scénario extrait de l'état de l'art afin de valider le modèle proposé.

Sommaire

5.1 Proposition méthodologique

Nous avons identifié dans les chapitres précédents le contexte de ce travail ainsi que les principales spécificités et problématiques propres au domaine étudié. Dans un souci d'amélioration et de résolution des problèmes de collaboration dans le PLM, nous avons identifié les besoins principaux pour répondre à la problématique de recherche « **définir une méthodologie qui permet de mettre en place une plate-forme Cloud Manufacturing et ainsi servir de support aux systèmes industriels traditionnels pour migrer vers le Cloud** » (cf. section 2.7) :

- pouvoir s'appuyer sur une démarche globale afin de fournir un guide aux utilisateurs pour migrer les systèmes classiques vers le cloud ;
- pouvoir modéliser le CM et formaliser les informations liées à ce domaine ;
- proposer une implémentation du CM afin de mettre en place une plate-forme et produire un outil d'aide à la décision.

Ce chapitre montre une adaptation de la méthodologie ASCI au domaine du CM, deux notions présentées dans les chapitres précédents. Nous définissons ainsi les méthodes et outils nécessaires aux différentes étapes de la méthodologie que nous appelons par la suite ASCI-Onto, puisqu'il s'agit d'une adaptation de la méthodologie ASCI aux ontologies comme le montre la figure 5.1. On y voit notamment que le modèle de domaine est une ontologie du CM.

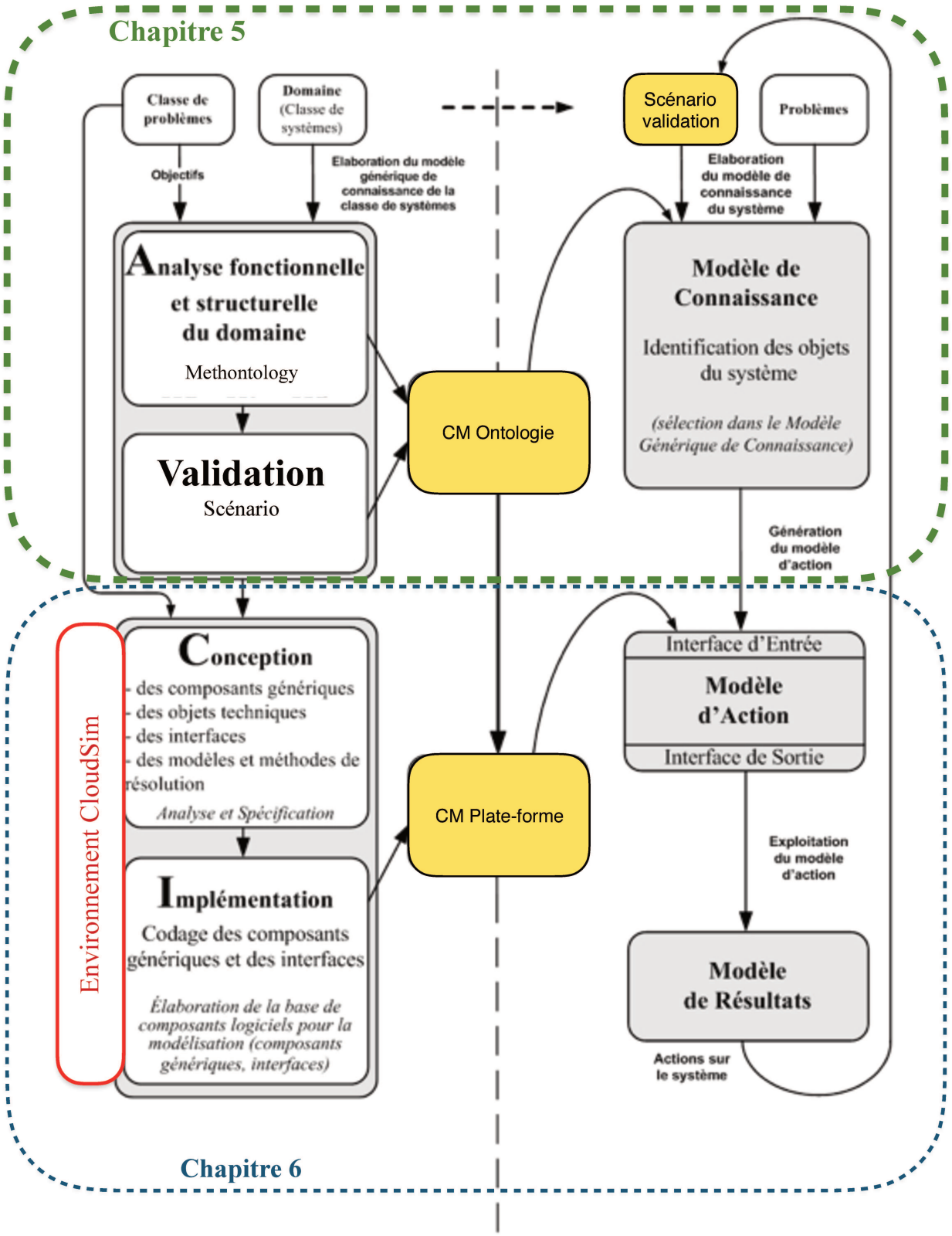


FIGURE 5.1 – *Positionnement du chapitre 5 dans la thèse*

Nous introduisons dans la section 5.2 la phase 1 préconisée par ASCI pour la création d'un modèle générique de connaissance appliqué au CM. Nous expliquons durant cette phase le choix des ontologies comme outil de modélisation et les étapes de production du modèle générique de connaissance. En effet, nous avons adapté les étapes préconisées par ASCI, et les avons remplacées par deux étapes adaptées aux ontologies : le développement de l'ontologie en suivant la méthodologie « methontology » et sa validation en utilisant la méthode « Ontoclean ».

5.2 PHASE 1 : La conception du modèle générique du domaine *Cloud Manufacturing* (CM)

5.2.1 Le modèle générique de connaissance du domaine dans ASCI

Lors de la première étape d'ASCI, l'objectif est de concevoir le modèle générique de connaissance du domaine qui est le résultat de deux étapes successives.

— **L'analyse fonctionnelle et structurelle du domaine** dont le but est de formaliser le domaine sous une forme graphique ou syntaxique. Durant cette étape, les différents experts du domaine et aussi de la modélisation travaillent ensemble pour rechercher et définir les entités qui composent le domaine, leurs fonctions ainsi que leurs associations. A ce niveau de la démarche, les objectifs de l'étude ainsi que les contraintes à respecter sont énoncés. Cette analyse donne une vue « statique » du domaine étudié qui est issue, si nécessaire, de la décomposition du système en trois sous-systèmes : sous-système physique, sous-système logique et sous-système décisionnel (section 4.3.2.1).

— **La spécification des entités et de leurs comportements** permet l'expression du fonctionnement du domaine et des flux présents dans le domaine. Durant cette étape, il est question de spécifier le système de pilotage de façon précise en prenant en compte les éléments qui le caractérisent comme par exemple les règles de gestion. Une fois réalisée, la spécification doit permettre aux experts en modélisation et aux experts du système de s'accorder sur le fonctionnement réel ou désiré du système.

À l'issue de ces deux étapes, le modèle générique de connaissance du domaine réalisé peut être instancié sur tous les système du domaine étudié. Dans les sections qui suivent, nous les différentes étapes de construction d'un modèle générique de connaissance pour le CM sont présentées. Elles sont adaptées aux étapes d'ASCI afin de fournir un modèle basé sur les ontologies.

5.2.2 Vers la construction d'un modèle générique de connaissance basé sur les ontologies

5.2.2.1 Vers une modélisation sémantique : Définitions

Nous avons expliqué dans la section 1.5.1 qu'il est nécessaire, en vue de réaliser l'interopérabilité *sémantique*, que les données échangées soient comprises par les systèmes. En effet, une donnée porteuse de sens devient une information qui peut être traitée par le système cible sans ambiguïté. Comme expliqué par [Fortineau 13a], Le terme « sémantique » est issu du grec $\sigma\eta\mu\alpha\iota\nu\omega$ (prononcé « sémaino ») qui veut dire : signifier.

Nous pouvons illustrer l'ambiguité sémantique par une recherche effectuée sur un moteur de recherche du type *Google* . La figure 5.2 montre un exemple d'une requête ambigue en utilisant le terme « chaîne » dans une requête soumise sur google où le moteur de recherche

renvoie toutes les réponses possibles pour le mot « chaîne » ainsi que les propositions du moteur de recherche pour la désambiguisation du sens du terme « chaine ».

FIGURE 5.2 – *Résultats de la requête « chaine » dans Google*

Il est important de distinguer la « syntaxe » de la « sémantique ». La syntaxe, particulièrement en programmation informatique, est liée à la forme, la structure ou la grammaire d'un langage. Elle permet de répondre à la question : comment construire une phrase ou une ligne d'un programme ? Comme toutes les langues, un programme informatique écrit dans un langage suit des règles qui définissent si oui ou non l'instruction est bien construite. C'est la syntaxe qui détermine si on doit utiliser des accolades ou des mots clés. Par exemple, dans le langage *Java*, une ligne doit se terminer par un point virgule et les variables doivent être déclarées avant leur utilisation dans le programme tandis que dans le langage *SQL* une requête de selection doit commencer par le mot clé **Select**.

La sémantique concerne le sens de la phrase ou l'instruction. Elle permet de répondre à la question : cette phrase est-elle valable ? Si oui, qu'est ce que la phrase veut dire ? Ainsi, pour un programme informatique, elle détermine sa signification.

[Fortineau 13a] explique que dans le cas de l'information numérique, s'intéresser à la sémantique signifie s'intéresser au sens que l'on donne aux données et que le traitement de la sémantique de l'information dans ce cas nécessite d'établir des liens sémantiques fondés sur le sens. Selon le même auteur, l'enjeu de la modélisation sémantique des informations implique que les termes utilisés aient la même signification pour chaque acteur.

Au début des années 2000, la modélisation basée sur les ontologies a vu le jour dans le cadre du web sémantique et est aujourd'hui une des voies les plus prometteuses pour répondre à cet enjeu.

Définition 5.1 *Le web sémantique n'est pas un nouveau web, mais l'extension de l'actuel, dans lequel les informations auraient un sens bien défini, permettant aux ordinateurs et aux utilisateurs de mieux travailler ensemble. Les premières étapes de tissage du web sémantique au travers de la structure actuelle du web sont déjà en cours. Dans un futur proche, ces développements vont ouvrir la voie à de nouvelles fonctionnalités, puisque les machines vont être bien plus capables de traiter et de « comprendre » les données, que, pour le moment, elles ne font qu'afficher [Berners-Lee 01].*

Le web sémantique repose largement sur les ontologies formelles qui structurent des données sous-jacentes afin que celles-ci soient compréhensibles et traitables par les machines (ordinateurs) sur le web [Roche 12]. Nous présentons plus en détail les ontologies comme outil de modélisation dans la section suivante.

5.2.3 Les ontologies : définition et langages

Gruber [Gruber 95] définit une ontologie comme suit :

Définition 5.2 *Une ontologie est une représentation explicite d'une conceptualisation [Gruber 95].*

En analysant cette définition, [Fortineau 13a] précise que *représentation* signifie qu'il s'agit d'un outil de modélisation ; *explicite* indique que l'ontologie est capable de modéliser l'ensemble des informations d'un domaine, quel que soit le niveau d'abstraction ou le type d'application et que *conceptualisation* indique que l'ontologie est de type concepts-propriétés et renvoie à la définition d'une sémantique propre au domaine. [Lee 09] définit une conceptualisation comme « une extraction du vocabulaire d'un domaine et une vue abstraite et simplifiée du monde que l'on souhaite représenter ».
La définition de[Gruber 95] a été légèrement modifiée par [Borst 97] :

Définition 5.3 *L'ontologie est une spécification formelle d'une conceptualisation partagée [Borst 97].*

Cette définition s'explique ainsi [Studer 98] : *explicite* signifie que le « type des concepts et les contraintes sur leurs utilisations sont explicitement définies », *formelle* se réfère au fait que « la spécification doit être lisible par une machine », *partagée* se rapporte à la notion selon laquelle une ontologie « capture la connaissance consensuelle, qui n'est pas propre à un individu mais validée par un groupe », *conceptualisation* se réfère à « un modèle abstrait d'un certain phénomène du monde reposant sur l'identification des concepts pertinents de ce phénomène ».
Les ontologies sont utilisées dans de nombreux domaines. [Guarino 98] a recensé ces domaines comme étant : l'ingénierie des connaissances, la modélisation qualitative, l'ingénierie des langages, la conception de bases de données, la recherche d'information, l'extraction d'information, la gestion et l'organisation de connaissance.

5.2.3.1 Notions sous-jacentes

[Maedche 01] propose une définition mathématique des ontologies :

Définition 5.4 *$O := C, R, H, ref, A^0$ où C représente l'ensemble des concepts (classes), disjoint de R qui représente l'ensemble des relation (propriétés). H représente la hiérarchie (taxonomie) entre les concepts, et ref, une fonction qui associe les relations non-taxonomiques aux différents concepts. Enfin A^0 est un ensemble d'axiomes exprimés dans un langage logique [Maedche 01].*

Concept (C)

Selon [Uschold 95] un concept représente, pour un objet matériel, une notion ou une idée. Il est composé de trois parties : un ou plusieurs **termes**, une **notion** et un **ensemble d'objets**. La **notion** représente la sémantique du concept et est définie à travers ses propriétés et ses attributs. La **notion** est appelée **intention** du concept. L'**ensemble d'objets** correspond aux objets définis par le concept qui sont les **instances.** Cet ensemble est appelé **extension**. Enfin les **termes** aussi appelés **labels** permettent de désigner le concept.

Relation sémantique (R)

[Hernandez 05] indique qu'une relation sémantique R est un type d'interaction entre les concepts d'un domaine $c_1, c_2, ..., c_n$. Cette dernière, se définit formellement à partir d'un produit de n concepts : $R : c_1 x c_2 x ... x c_2$. Il existe selon [Hernandez 05] deux types principaux de relations sémantiques.

— Relation taxonomique (H) : aussi appelée relation « est un » ou relation de spécificité/-généricité ou encore relation de subsomption est une relation binaire particulière qui permet d'organiser hiérarchiquement un ensemble de concepts. Elle implique l'engagement sémantique suivant [Guarino 02] : un concept c_1 subsume un concept c_2 si le concept c_1 est plus général que le concept c_2. Les instances se rapportant au concept c_2 sont des instances de c_1, en revanche une partie seulement des instances de c_1 seront des instances de c_2. C'est une relation dite d'ordre partiel définie à partir des propriétés suivantes :
  - l'asymétrie : si une classe d'individus X est incluse dans une classe d'individus Y alors Y n'est pas incluse dans X ;
  - la transitivité : si la classe d'individus X subsume une classe Y, qui elle-même subsume Z alors X subsume Z ;
  - la non réflexivité : non (X subsume X).

  Les ontologies ont la capacité de gérer l'**héritage multiple**, qui est une propriété selon laquelle un concept d'une ontologies peut avoir plusieurs pères par la relation de subsomption et hérite des propriétés de tous ses pères.

— Relation associative (ref) : il s'agit de relation d'interaction entre deux concepts autre que la relation de subsomption. Elle correspond à la notion de rôle en logique de description et permet de typer les liens entre concepts. Ces relations sont soit des propriétés entre concepts, soit entre concepts et attributs dans le cas où elles associent un concept et un type de données.

Axiome (A)

Les axiomes ont pour objectif de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ce sont des expressions qui sont toujours vraies et leur inclusion dans une ontologie peut avoir plusieurs objectifs

[Hernandez 05] : définir la signification des composants, définir les restrictions sur la valeur des attributs, définir les arguments d'une relation, vérifier la validité des informations spécifiées ou en déduire de nouvelles.

5.2.3.2 Types d'ontologies

La classification décrite dans [van Heijst 97] distingue trois types d'ontologies selon la structure des informations qu'elles contiennent.

- **Les ontologies terminologiques ou linguistiques** spécifient les termes utilisés pour représenter la connaissance d'un domaine. Un exemple de ce type d'ontologie est le réseau sémantique UMLS (Unified Medical Language System) [Lindberg 93].
- **Les ontologies de l'information** spécifient la structure des enregistrements d'une base de données. Les schémas de base de données en sont un exemple. Elles proposent un cadre de représentation de l'information stockée mais ne spécifient pas de détails sur la sémantique des champs.
- **Les ontologies pour la modélisation de la connaissance** spécifient la conceptualisation de la connaissance. Ces ontologies ont une structure beaucoup plus riche que celle des deux autres types. Elles sont généralement conçues en fonction de l'utilisation prévue de la connaissance qu'elles contiennent.

Une autre classification, de type informatique et basée sur le type de données que les ontologies permettent de manipuler, a été proposée par [Fankam 08]. Ces deux types sont [Fankam 08] :

- **les ontologies de stockage** qui ne traitent que des données canoniques, c'est-à-dire qu'une instance ne peut se trouver que dans une et une seule classe à un moment donné. Ces ontologies permettent une description claire et structurée du domaine métier et sont adaptées à des problématiques de type « catalogue ».
- **Ontologies d'inférence** permettent de traiter les données non-canoniques, c'est-à-dire qu'une instance peut se trouver dans plusieurs classes en même temps. L'avantage de ces ontologies et leur capacité d'inférer des informations : pouvoir déduire des appartenances d'instance à des classes grâce aux propriétés, axiomes et restrictions explicitement définis dans l'ontologie.

Un autre critère pour la classification des ontologies est leur contenu, c'est-à-dire le sujet de la conceptualisation [van Heijst 97], [Hernandez 05].

- **Les ontologies génériques** définissent des concepts considérés comme génériques à plusieurs domaines. L'ontologie WordNet [Miller 98] est un exemple d'une telle ontologie dont le but est de représenter la langue naturelle anglaise. WordNet est composée d'ensembles de synonymes appelés *synsets*, où chaque terme est regroupé en classe d'équivalence sémantique, ainsi chaque ensemble de synonymes représente un concept particulier [Hernandez 05]. Selon [Charlet 02] la limite de ces ontologies est leur difficile réutilisation car leur objectif est de recouvrir tous les sens des mots et elles ne normalisent pas le sens de ces termes.
- **Les ontologies de domaine** sont des conceptualisation spécifiques à un domaine particulier. Ces ontologies posent des contraintes sur la structure et le contenu de la connaissance du domaine. A la différence des ontologies génériques, ces ontologies permettent d'organiser au sein d'un modèle conceptuel des connaissances, à partir de la com-

préhension du domaine et de l'application visée. par conséquent elles ont l'avantage d'offrir la possibilité de normaliser les concepts.
— **Les ontologies d'application** contiennent toutes les définitions nécessaires pour modéliser la connaissance propre à l'élaboration d'une tâche particulière. Ces ontologies sont rarement réutilisables pour une autre application.
— **Les ontologies de représentation de la connaissance** proposent un cadre de représentation sans émettre d'hypothèse sur le monde. Elles sont désignées comme ontologies abstraites ou de haut niveau parce qu'elles permettent de définir des concepts abstraits et peuvent être re-utilisées pour définir des concepts spécifiques.

Dans ce travail, nous utilisons les ontologies de domaine munies de la capacité d'inférence pour modéliser notre domaine pour les raisons explicitées dans la section 5.3.2.

5.2.3.3 Les langages de représentation d'ontologie

Nous présentons dans cette section les langages de spécification d'ontologie. Ces langages sont orientés Web sémantique et ont été pour la plupart recommandés par le W3C [1].

Logique descriptive repose sur trois notions : les concepts représentant des classes qui sont les objets, les rôles qui représentent les relations liant deux objets et les individus qui sont les instances des classes. Les logiques de description divisent la connaissance en deux parties [Kontchakov 10, Baader 91] (i) les informations terminologiques (Tbox) qui représentent l'ensemble des concepts génériques et leurs relation et (ii) les informations sur les individus (Abox) ou boîte assertionnelle qui est constituée des individus et de leurs descriptions. Par analogie avec les bases de données, la Tbox correspond à l'**intention** qui représente la structure de la base de données aussi appelée schéma, et la Abox à l'**extension** qui désigne l'ensemble de données organisées selon l'intention.

***Extensible Markup Language* (XML)** est un langage informatique à balises pour la structuration de données et de documents. La syntaxe XML est extensible car elle permet de définir plusieurs extensions avec chacune leur vocabulaire et grammaire comme par exemple : XHTML pour les pages web, XSLT pour la transformation des fichiers XML vers un autre format. XML-schéma [Fallside 2001] permet de définir la structure, les contraintes, et la sémantique de documents XML.

***Resource Description Framework* (RDF)** permet d'encoder, d'échanger et de réutiliser des méta-données structurées [Lassila 99]. C'est un langage destiné à décrire de façon formelle les ressources web ainsi que leurs propriétés et états afin de permettre le traitement automatique de telles descriptions. RDF-Schéma définit les relations entre ces ressources. Le pouvoir sémantique de ces deux langages est limité car les axiomes ne peuvent pas être directement décrits et le type de relations (symétrique, transitive,...) ne peut être spécifié [Hernandez 05].

DAML+OIL est conçu pour décrire la structure d'un domaine, basé sur approche orientée objet, il décrit la structure en termes de classes et propriétés [Horrocks 02]. D'un point de vue formel, DAML+OIL repose sur RDF et RDF-schema et fournit en plus des primitives plus riches issues de la logique descriptive. Le langage est mieux adapté que RDF à l'utilisation

1. https://www.w3.org consulté en Janvier 2015

et la maintenance d'ontologies mais présente des limites quant à la construction d'ontologies [Bechhofer 01].

***Ontologie Web Language* (OWL)** est le standard actuel proposé par le W3c pour représenter les ontologies. OWL se veut plus représentatif du contenu du Web que XML, RDF et RDF-Schema en apportant un nouveau vocabulaire avec une sémantique formelle [Hernandez 05]. Il a été créé pour être utilisé par les applications cherchant à traiter le contenu de l'information et non plus uniquement à présenter l'information. OWL est décliné en trois sous-langages d'expressivité croissante : OWL-Lite, OWL DL et OWLFull.

OWL-Lite est simple, facile à programmer avec un raisonnement rapide grâce à une expressivité limitée, à des hiérarchies et des contraintes simples. OWL DL possède une expressivité élevée tout en garantissant une complétude de calcul et de décidabilité nécessaires aux systèmes de raisonnement. La complétude de calcul assure que toutes les inférences seront prises en compte et la décidabilité assure que tous les calculs seront terminés dans un temps fini. OWL Full a une expressivité maximale avec une liberté de syntaxe RDF. Les raisonnements en OWL Full sont souvent très complexes, lents, incomplets et indécidables.

5.3 Proposition

5.3.1 Modèles du CM : état de l'art

Nous avons réalisé un état de l'art des modèles et ontologies du CM [Talhi 14] dont le but était de trouver dans la littérature un modèle qui nous permet de représenter notre domaine et qui servira comme référence pour implémenter et de mettre en place une plate-forme CM en s'appuyant sur le cadre architectural proposé par [Xu 12b] (figure 3.9). Nous avons montré [Talhi 14] qu'il n'existe pas de modèle générique et complet du CM ; la généricité et la complétude de ces modèles étant basées sur la capacité de ces derniers à couvrir toutes les couches du cadre architectural proposé par [Xu 12b]. En effet, les modèles recensés durant l'état de l'art se focalisent sur la couche *Manufacturing Resource Layer* (MRL) afin de proposer une modélisation des ressources et opérateurs au sein des ateliers. Par conséquent, ces travaux se positionnent au niveau du fournisseur de ressources et leurs objectifs sont de proposer des méthodes d'intégration des connaissances dans les systèmes manufacturiers.

Par exemple, [Vincent Wang 13] proposent une modélisation des services manufacturiers utilisant EXPRESS-G. Leur modèle décrit le service, le fournisseur et les requêtes des utilisateurs. EXPRESS-G est un formalisme graphique du langage EXPRESS qui a été défini pour donner une vue synthétique des problèmes à modéliser [Paviot 11]. Malheureusement, la modélisation de données en EXPRESS-G n'est pas idéale pour le traitement de données hétérogènes provenant de diverses ressources sur Internet et par conséquent ne permet pas l'interopérabilité de ces différente ressources. [Lu 14] utilisent une ontologie afin de faciliter la gestion et l'accès des utilisateurs aux services dans une plate-forme cloud. L'ontologie est utilisée pour créer des instances d'utilisateurs et de fournisseurs de services en mettant l'accent sur la gestion des règles d'accès correspondantes.

[Ameri 12] introduisent une méthodologie pour développer des ontologies dont le but est de représenter les capacités manufacturières. Ces ontologies permettent de définir les usines aussi bien que les ressources et les opérateurs. [Lin 11] utilisent une ontologie pour modéliser les ressources d'un atelier et [Jiang 10] présentent un cadre de modélisation qui permet l'intégration des connaissances dans un réseau manufacturier collaboratif. Dans ces études, les

auteurs proposent des solutions du point de vue du fournisseur, car ils utilisent des ontologies pour modéliser le fonctionnement interne des ateliers et des usines.

Puisque dans le CM, tout est considéré comme un service, cela signifie que la partie « production » est considérée comme un service aussi bien que toutes les autres phases du cycle de vie. Les travaux présentées ci-dessus traitent le CM du point de vue du fournisseur en proposant des solutions permettant de modéliser les usines, ateliers et opérateurs. Afin d'implémenter l'architecture CM il est nécessaire de traiter ce domaine en nous positionnant au niveau du GSL. En effet, c'est dans la couche GSL que les services proposés par les fournisseurs sont identifiés, localisés et attribués aux utilisateurs.

De plus, nous avons montré dans le chapitre 3que le CM est un domaine qui était peu étudié au début de ces travaux. Par conséquent, il nous était nécessaire d'étudier les modèles ontologiques issus du CC. [Brandis 14] introduisent une modèle permettant la gouvernance des cloud en traitant l'aspect sémantique. [Funika 13] présentent une approche basée sur les ontologies afin de développer un outil de monitoring des applications développées dans des environnements cloud. [Chang 12] proposent un médiateur orienté agent qui mémorise le contexte de l'utilisateur et lui propose des services adéquat dans un cloud mobile. [Liu 14] proposent une solution permettant de contrôler les services dans un cloud et de gérer l'accès des utilisateurs. [Kourtesis 14] discutent l'état actuel et les enjeux de la gestion de la QoS dans le cloud. Enfin, [Moscato 11] présentent une plate-forme appelé MOSAIC, qui utilise les ontologies pour fournir une description unifiée des composants d'un système CC afin d'aider les développeurs à déployer des applications sur plusieurs clouds communiquant entre eux.

5.3.2 L'apport des ontologies à la modélisation du domaine

Le choix d'une modélisation ontologique est motivé par les raisons suivantes :

— Comme expliqué dans les sections précédentes, nous avons choisi les ontologies comme outil de modélisation car elles proposent une une vue simplifiée et compréhensible du domaine et elles permettent de représenter explicitement le sens des concepts avec leurs relations. Dans ce travail, nous proposons une ontologie d'inférence afin de nous permettre de faire des déductions sur les instances concernant les classes, les propriétés et axiomes définis dans l'ontologie [Krima 09].
— L'ontologie a pour but de représenter notre domaine d'étude qui est le CM. L'enjeu ici est d'unifier les vues multiples de ce même domaine car les fournisseurs n'utilisent pas les mêmes termes pour décrire leur services. [Moscato 11] précisent que les différents systèmes cloud et fournisseurs possèdent différentes méthodes pour invoquer les services, spécifier les besoins en termes de services et donc utilisent des termes variés pour décrire les mêmes services. Comme expliqué dans l'article de [Kabmala 06] « les ontologies ont été proposées pour résoudre les problèmes qui résultent de l'utilisation de différentes terminologies pour désigner le même concept ou de l'utilisation d'un même terme pour désigner des concepts différents ». Par conséquent, la définition d'une ontologie de CM nous permet de remédier à ce problème car elle contient tous les concepts communs liés au CM.
— La gestion des données non-canoniques est une caractéristique de certaines ontologies qui permet à une instance d'appartenir à plusieurs classes. Ceci est important dans notre cas car, par exemple, un logiciel de conception représenté par une instance X doit appartenir à toutes les classes représentant les logiciels de conception : « Software as a

service », « Design as a service », « Design on line », etc. Cela permet d'intégrer l'aspect multi-vues à la modélisation.

— L'ontologie sera exploitée au sein d'une plate-forme CM afin d'améliorer la découverte de services par les utilisateurs. Cette amélioration consiste à rendre la découverte de services « intelligente » et « automatique ».
  - l'intelligence ici veut dire dire que l'ontologie doit permettre la découverte, l'identification et l'invocation du service adéquat grâce à des déductions logiques ce qui réduit l'engagement de modélisation et facilite la maintenance du modèle ;
  - l'automatisation signifie que le processus de découverte de service est exécuté sans intervention manuelle de l'utilisateur. Selon [Tsai 08], les capacités de raisonnement fournies par les ontologies facilitent le processus de mappage de services dans des environnements orientés services, et offrent un certain niveau de flexibilité en renvoyant comme réponse les services qui sont compatibles avec la demande de l'utilisateur.

  Cependant, les deux notions sont étroitement liées car l'« automatisation » peut être réalisée grâce à l'« intelligence » du modèle. En effet, le mappage sémantique entre l'ontologie du CM et la description des fournisseurs facilite le processus de découverte sans effort de la part de l'utilisateur. Ce mécanisme est basé sur le partage d'une sémantique commune par les fournisseurs. Par conséquent, au lieu de contraindre les fournisseurs d'utiliser les mêmes termes pour décrire leurs services, ces derniers partagent le sens de ces termes. Ainsi, en reprenant l'exemple ci-dessus du service de conception représenté par des termes (concepts) différents selon le fournisseur, l'important est que pour chaque utilisateur cela représente un logiciel de conception. Pour réaliser ceci, il doit exister un partage minimum des concepts de l'ontologie afin que les informations soit comprises de la même manière par tous les utilisateurs.

5.4 Proposition d'un modèle générique de connaissance basé sur les ontologies

5.4.1 Langage utilisé

Selon [Antoniou 04], un langage d'ontologie doit respecter les besoins suivants :

1. une syntaxe bien définie ;
2. une sémantique bien définie ;
3. supporte le raisonnement ;
4. expressivité et capacité d'expression.

Une syntaxe bien définie est une condition nécessaire pour le traitement de l'information par la machine, alors qu'une sémantique formelle permet de donner du sens aux données. La sémantique est une condition préalable au raisonnement où des déductions peuvent être faites sur les données automatiquement. la capacité d'expression est fournie par l'expressivité du langage. Nous avons ainsi choisi le langage *Web Ontology Language* (OWL) pour représenter notre ontologie et avons utilisé le logiciel Protégé pour la développer.

Nous l'avons introduit auparavant, OWL est le plus récent des langages d'ontologies [Antoniou 04]. Il est composé que trois catégories de langages : OWL-Lite, OWL-DL et OWL-Full. Ces langages différent dans leur niveau d'expressivité : OWL-full etant le plus expressif. D'une manière générale, plus le langage est riche et permet une grande expressivité, moins

le raisonnement est efficace, puisque cela augmente sa non-décidabilité. Ainsi, nous avons besoin d'un langage qui peut être supporté par les raisonneurs tout en étant suffisamment expressif pour pouvoir représenter notre domaine. A cet effet, nous utilisons OWL-DL, car il représente un compromis en termes d'expressivité et de raisonnement.

5.4.2 Méthodologies de construction et validation de l'ontologie CM

Développer un modèle pour un domaine donné n'est jamais une tâche facile du fait qu'il doit être aussi exhaustif et générique que possible. Cela veut dire que les concepts couverts par l'ontologie doivent être compris par toutes les parties prenantes. [López 99] proposent une méthodologie expliquant les étapes a suivre afin de développer une ontologie, il s'agit de la méthodologie METHONTOLOGY. Cette méthodologie s'applique à clarifier les différentes étapes de la construction de l'ontologie en respectant des activités de gestion de projets (planification, assurance, qualité), de développement (spécification, conceptualisation, formalisation, implémentation, maintenance) et des activités de support (intégration, évaluation, documentation). Les différentes étapes proposées sont listées ci-après [López 99, Fernández-López 97] .

1. *Spécification* : permet de produire un document de spécification de la future ontologie. Il décrit entre autres l'objet de l'ontologie, ses utilisateurs et utilisations ainsi que le degré de formalisation à employer.
2. *Acquisition de connaissance* : mène à l'identification des termes de l'ontologie et leur définition. Des réunions de brainstorming, des interviews d'experts et des analyses de textes sont des techniques d'acquisition de connaissance susceptible d'être utilisées.
3. *Conceptualisation* : vise à structurer la connaissance du domaine en un modèle conceptuel. La représentation est à ce stade informelle.
4. *Intégration* : permet d'envisager quelles sont les ontologies existantes qui pourraient être intégrées dans la construction de l'ontologie.
5. *Implantation* : cette étape consiste à représenter formellement l'ontologie à partir d'un langage choisi.
6. *Evaluation* : permet de vérifier et valider l'ontologie en fonction de son environnement logiciel et sa documentation. Il est ainsi question de vérifier les problèmes de cohérence, d'incomplétude et de répétition.
7. *Documentation* : les documentations permettent la compréhension globale de l'ontologie et sa réutilisation. Les auteurs précisent que les documentations sont généralement incomplètes et proposent de pallier ce manque en imposant la rédaction de documentation à la fin de chacune des phases de la construction de l'ontologie.

Selon [Hernandez 05], la particularité de la méthodologie est de s'attacher fortement à la maintenance de l'ontologie de domaine et à son évaluation. Les auteurs de METHONTOLOGY insistent sur le fait que les concepteurs doivent s'efforcer autant que possible d'utiliser des ontologies existantes. Nous nous appuyons sur cette méthodologie ainsi que les modèles pour le CM présentés dans la section précédente pour concevoir notre ontologie de domaine *Cloud Manufacturing Ontology* (CMO).
La méthodologie OntoClean [Guarino 09] est un autre type de méthodologie visant à permettre la validation d'une ontologie. Elle a été proposée afin de vérifier qu'il n'y a pas d'incohérence « philosophique » dans la description des classes et relations. Elle repose sur la

définition de caractéristiques (aussi appelées méta-propriétés) des concepts pour structurer des ontologies en imposant certaines contraintes sur l'utilisation des liens de subsomption [Guarino 00] :

— l'identité (I) : un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept. Par exemple, le concept « étudiant » porte une propriété d'identité liée au *numero d'étudiant*, deux étudiants étant identiques s'ils ont le même numéro ;
— l'unité (U) : exprime le fait que les instances d'un concept sont un ensemble délimité. Par exemple le concept « océan » est unité car ses instances telles que « océan atlantique » sont uniques par contre le concept « eau » n'est pas unité car n'est pas délimité ;
— la rigidité (R) : une propriété d'un concept est rigide si elle est essentielle pour chacune des instances du concept, c'est-à-dire que chacune des instances détient cette propriété pour exister. La propriété « être un humain » par exemple, est rigide mais « être étudiant » ne l'est pas ;
— la dépendance (D) : deux concepts sont dépendants quand ils partagent des instances, autrement dit, un concept C_1 est dépendant d'un concept C_2 si, pour toute instance de C_1, il existe une instance de C_2 qui ne soit ni partie ni constituant de l'instance de C_1.

De plus, ces quatre méta-propriété sont valorisées [Fortineau 13a] : elles peuvent être positives (« + », toutes les instances correspondent à la méta-propriété), négatives (« - » toutes les instances ne correspondent pas nécessairement à la méta-propriété) ou anti- (« $\sim$ », aucune instance ne correspond à la méta-propriété) Ainsi, à partir de ces propriétés, il est possible d'établir des règles de conception, fondées sur la subsomption (hiérarchie) entre propriétés, notamment, dans le cas ou p est sous-classe de q (q subsome p) :

— si q est anti-rigide ($\sim$R), alors p l'est aussi ;
— si q porte un axiome d'identité (+I) alors p en hérite ;
— si q porte l'unité (+U) alors p en hérite ;
— si q porte l'anti-unité ($\sim$ U) alors p également ;
— toute instance doit appartenir à une seule sur-propriété +I.

D'autre part, selon ASCI, le modèle générique de connaissance regroupe tous les systèmes du domaine étudié en identifiant leurs points communs. Simon [Simon 62] définit un système complexe comme un système constitué d'un grand nombre de composants qui interagissent d'une manière non-triviale. Dans de tels systèmes, compte tenu des propriétés des composants et des lois qui gouvernent leurs interactions, il est difficile d'en déduire les propriétés du système global. [Paviot 10] argue que c'est la non-trivialité des interactions des acteurs dans un système qui le rend complexe et non pas le fait qu'il soit de très grande taille.

Nous considérons ainsi un système CM comme un système complexe dès lors que ses composants sont fortement connectés. Le CM met en effet en correspondance un ensemble d'acteurs (fournisseurs et clients) qui détiennent ou utilisent des ressources virtualisées et distribuées. De plus, chaque ressource peut être virtualisée en plusieurs services utilisés par différents clients. Selon la décomposition de Le Moigne [Le Moigne 92], notre ontologie comprend le SSP et le SSL car elle décrit les différentes ressources et service du système. Le SSD n'est pas inclus dans l'ontologie car celle-ci doit être la plus générique possible alors que les règles de gestion et de décision sont propres et internes à chaque entreprise. En effet, une règle de gestion peut concerner la décision que prend une entreprise quant à la visibilité

et l'accès à ses services. Cette même entreprise peut restreindre l'accès aux concurrents. Par conséquent, le choix des règles n'est pas toujours le même et n'a pas toujours le même niveau de granularité pour toutes les entités du système.

5.4.3 L'ontologie CM comme une base de connaissance : concepts génériques

Les concepts racines ou les concepts de haut niveau de la *Cloud Manufacturing Ontology* (CMO) sont illustrés par la figure 5.7 [Talhi 14]. Pour la conception de cette ontologie, nous avons suivi la méthodologie METHONTOLOGY en intégrant des concepts identifiés dans les modèles recensés durant l'état de l'art et présentés dans la section 3.5 tout en adaptant le tout au domaine du CM. Selon [Fortineau 13b], la définition des concepts racines de l'ontologie doit respecter les règles décrites ci-dessous et que nous avons prises en compte lors de la définition des concepts de notre ontologie.

- les concepts racines doivent être sémantiquement disjoints. Cela signifie qu'il doit être possible de spécialiser les concepts racines sans ambiguïté d'appartenance : une classe spécialisée ne saurait être sous-classe de deux concepts racine différents ;
- la sémantique de l'ensemble des concepts racines doit être exhaustive. Ils doivent permettre de représenter l'ensemble des informations requises pour la modélisation des produits ;
- les concepts racine doivent être suffisamment génériques pour parler aussi bien aux ontologies de domaine qu'aux applications métier. Ils doivent donc s'affranchir du domaine d'application [Lee 09], mais être tout de même porteur de sens, même aux niveaux applicatifs les plus bas [Fiorentini 10] ;
- enfin les concepts racines doivent être porteurs de sens pour l'ensemble des acteurs concernés [Lee 10].

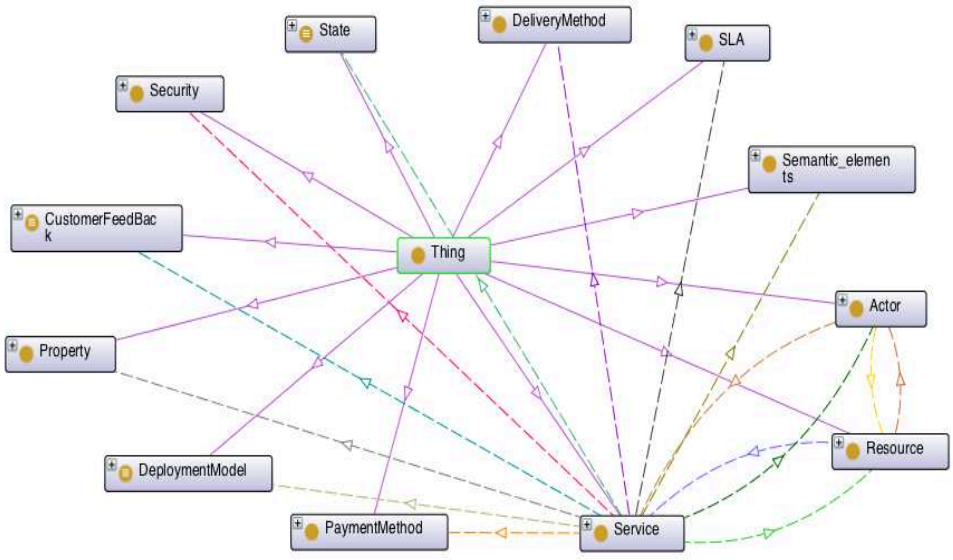


FIGURE 5.3 – *Top level of CMO*

L'ontologie sert en premier lieu de base de connaissance pour décrire le domaine CM. Ci-dessous, nous décrivons les classes et sous classes de la CMO que nous avons jugées per-

tinentes. Notre ontologie n'est pas exhaustive, ainsi d'autre concepts peuvent être ajoutés. Chaque concept racine comporte un ensemble de sous-classes qui sont liées à ces dernier via un lien taxonomique ou sémantique. La convention typographique utilisée dans ce qui suit est la suivante : **Classe**, instance et *propriété*. Les concepts racines sont les suivants :

— **Actor** représente les acteurs qui interagissent avec le système CM. Ce concept se divise en un ensemble de sous-classes : **Person** qui représente les utilisateurs particuliers et **Enterprise** pour les utilisateurs professionnels. La classe **Actor** contient également les sous-classes : **consumer** pour représenter les clients et **Provider** pour les fournisseurs de service. Contrairement aux sous-classes : **Person** et **Enterprise**, **Consumer** et **Provider** ne sont pas disjointes, car un acteur du système peut être à la fois fournisseur et consommateur de services. Par exemple, une entreprise peut fournir un ensemble de services, mais aussi être identifiée en tant que cliente pour des services Cloud comme dans le cas de l'externalisation ou de la sous-traitance.

— **DeploymentModel** permet de représenter les méthodes de déploiement du CM. Ces modes ont été présentés dans le chapitre 1 et contient les instances suivantes : Community, Hybrid, Private, Public. Ces modes sont représentés par des instances et non pas des sous-classes car on arrive au niveau de granularité le plus bas, c'est-à-dire a l'entité élémentaire de la taxonomie ainsi Community peut être associé à toute instance de cloud pour indiquer qu'il est communautaire. Il n'y a pas besoin d'avoir Community1, Community2, ..., CommunityN car la définition d'un cloud communautaire est unique. Ces instances (Community, Hybrid, Private, Public) seront associées aux instances de la classe **Service** pour décrire leur visibilité.La classe **DeploymentModel** est donc une classe énumérée.

— La classe **Semantic_elements** contient tous les éléments nécessaires pour permettre la communication avec les services et faciliter l'utilisation de ces derniers. Elle se compose des sous-classes suivantes : **Data_language, Platform_language, Protocol**. La sous-classe **Data_language** représente le format d'échange des données en entrée et sortie. Nous avons considéré les formats suivants : step et xml, qui sont des instances de cette classe, sachant qu'il est toujours possible de peupler l'ontologie avec d'autres formats. **Platform_language** contient les langages de programmation (comme java, php, sql, python) utilisés par les plates-formes dans le cas d'un Paas ainsi que les environnement d'exécution adéquats. Finalement la sous-classe **Protocol** contient les protocoles utilisés par les services comme par exemple HTTP [2], SMTP [3], POP3 [4], IMAP4 [5], et le protocole FTP [6].

— **SLA** contient les éléments nécessaires pour définir le SLA. Comme nous l'avons présenté précédemment (section 1.5.3), le SLA est un document qui spécifie le niveau de service ainsi que toutes les garanties assurés par le fournisseur, et la procédure de dédommagement si le service ne respecte pas ces engagements. En conséquence, le SLA est défini pour des services payants [Bradshaw 10]. Cette classe contient les sous-classes suivantes : **Price, Metric, Initiation_time** et **Recovery_time**. La sous-classe **Price** définit le prix des différents services, **Metric** contient des éléments expliquant comment le service peut être mesuré et contient les concepts suivants :

2. http://www.w3.org/Protocols/rfc2616/rfc2616.html consulté en Octobre 2015
3. https://www.ietf.org/rfc/rfc2821.txt consulté en Octobre 2015
4. https://www.ietf.org/rfc/rfc1939.txt consulté en Octobre 2015
5. http://tools.ietf.org/html/rfc3501 consulté en Octobre 2015
6. http://www.w3.org/Protocols/rfc959/ consulté en Septembre 2015

— **Availability_Rate_Metric** : le pourcentage de disponibilité des services, mesuré par la disponibilité totale sur le temps total ;
— **Outage_Duration_Metric** : la durée d'une panne, mesurée par le temps entre le début et la fin de la panne et exprimée en heures et minutes ;
— **Mean_Time_Between_Failures_Metric** : le temps attendu entre deux défaillances consécutives, mesuré par la durée de fonctionnement et le nombre de défaillances, exprimé en nombre moyen de jours ;
— **Reliability_Rate_Metric** : le pourcentage des résultats réussis après exécution des services, mesuré par le nombre total de réponses réussies exprimé en pourcentage ;
— **Network_Capacity_Metric** : les caractéristiques mesurables de la capacité du réseau, mesurées en termes de bande passante[7] exprimé en Méga-bits par seconde.

La sous-classe **Initiation_time** décrit les temps nécessaire pour configurer une machine virtuelle ou physique tandis que que **Recovery_time** représente le temps nécessaire de récupération après pannes .

— **State** décrit l'état d'un service et contient les individus suivants : free pour exprimer que le service est libre et donc peut être alloué, unavailable, pour un service indisponible comme par exemple durant la maintenance des ressources ou used pour un service en cours d'utilisation.
— La classe **Property** (Figure 5.4) est divisée en deux sous-classes : **functional_property** et **non_functional_property**. La classe **functional_property** décrit les éléments nécessaires à l'exécution d'un service. Elle contient :

— **Manufacturing_resource_configuration** qui définit les notions nécessaires à la description d'une ressource manufacturière comme dans le cas d'une fraiseuse par exemple. Ces notions sont représentées par les concepts suivants : **material, part_size, resource_type**,
— La classe **Storage_capacity** permet de définir la capacité de stockage d'une ressource, elle est liée au domaine d'application (range) par la propriété *storage_capacity_value*. Nous pouvons illustré ceci par l'exemple de Google Drive[8] ou l'utilisateur peut choisir l'espace de stockage adéquat,
— **VM_configuration** décrit les caractéristiques d'une machine virtuelle utilisée dans le modèle de service Iaas et contient : **VM_CPU, VM_OS, VM_RAM, VM_storage_capacity**.

7. Le terme bande passante est utilisé pour désigner le débit binaire maximal d'un canal de communication.
8. https://www.google.com/intl/fr_fr/drive/using-drive/#start consulté en Octobre 2015

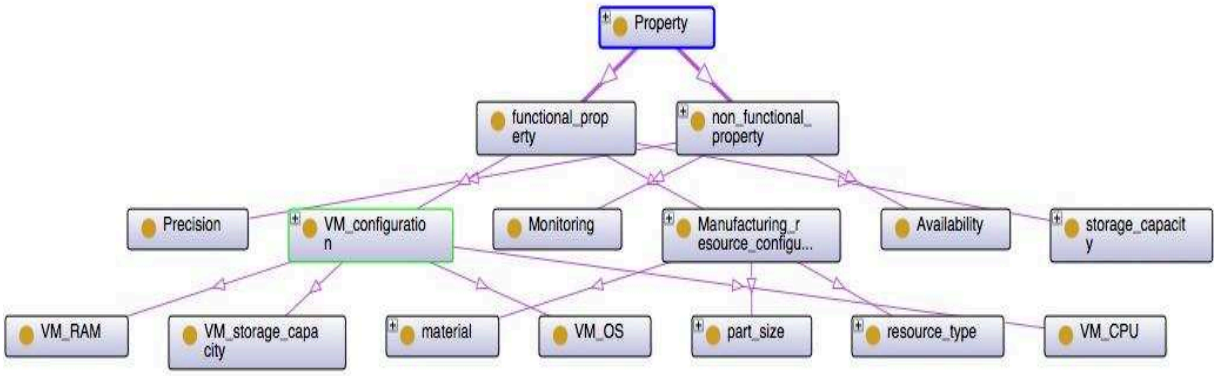


FIGURE 5.4 – *Taxonomie du concept* ***Property***

La classe **Non_functional_propety** quant à elle englobe les propriétés qui sont à considérer, en plus des propriétés fonctionnelles, pour décider quel service est le mieux adapté pour un utilisateur particulier comme par exemple : **Availability** pour décrire la disponibilité du service, la **Qos, Reliability**, pour qualifier le service et finalement **Traceability** dans le cas où le service permet la traçabilité des opérations...

— **Security** contient les éléments permettant la représentation des méthodes de sécurité utilisées dans le domaine CM. La sécurité est considérée comme étant un point important et crucial à l'adoption du cloud [Dukaric 13]. [Dukaric 13] propose une décomposition de la couche « sécurité » dans le CC en : authentification, autorisation, les groupes de sécurité, single sign-on et le monitoring. En nous basant sur cette définition nous avons gardé **authentication, authorization, security groups, single sign-on** comme sous classes de la classe **Security** car nous estimons que le monitoring est une propriété non fonctionnelle qui peut être considérée lors de l'utilisation du service.
— **Resource** décrit la ressource qui sera virtualisée en tant que service. Une ressource manufacturière est définie par [Vincent Wang 13] comme étant : capacités manufacturières matérielles et non-matérielles comme les équipements, les machines, l'intelligence et l'expertise. Notre classe **Resource** contient deux sous-classes : (i) **Hard_resource** pour représenter les ressources matérielles : **Manufacturing_equipment** and **Computational_resources** et (ii) **Soft_resource** qui décrit les ressources non-matérielles : **Software, Experience, Skill, knowledge**. En effet, un service peut représenter une application et la formation associée.
— **Service** est l'entité centrale d'un système CM. Dans le CM toutes les ressources sont virtualisées en tant que service. Cette classe contient les phases du PLM présentées comme des services : **Design_aaS, Simulation_aaS, Experimentation_aaS, Manufacturing_aaS, Maintenance_aaS** et les trois modèles de services du CC **Saas, Paas, Iaas**

(la relation entre le CM et le CC est discutée dans la section 3.3). Nous avons ajouté le modèle de service **Holonaas** où les ressources physiques comme les machines-outils sont fournies à l'utilisateur comme un service. Le concept de « *Holon* » représentant une entité autonome composée d'une partie informationnelle et une partie physique (section 2.3). [Wu 13b] explique que les fournisseurs de ressources physiques sont responsables de l'exploitation des équipements de fabrication, des technologies d'usinage et de finition ainsi que les technologies d'inspection et d'emballage de ressources. Les systèmes holoniques permettent dans cette situation de modéliser et d'assurer l'accès à ces ressources physiques mais aussi à des usines entières via Internet.
Nous avons utilisé des axiomes pour définir les classes représentant les phases du PLM, comme nous l'avons vu précédemment, les axiomes permettent de restreindre un concept ou d'y associer des instances automatiquement. Par exemple la classe : **Design_aaS** ≡ Service and (encompasses_resource *some* (Design_knowledge or Design_skills or Desing_software))

— **PaymentMethod** et **DeliveryMethod** sont deux classes qui contiennent des individus qui seront associés au **Service** pour spécifier le moyen de paiement de l'utilisateur et le mode de livraison du service. Elles sont constituées respectivement des sous-classes sous-classes suivantes : **Cheque, CreditAccount, CreditCard** et **AirTransport, Online, SeaTransport, SurfaceTransport**.
— **CustomerFeedback** fait référence à l'évaluation du service émise par le client après utilisation de celui-ci. Elle se compose des individus suivants : average, outstanding, satisfactory, unsatisfactory, very_good .
Les liens hiérarchiques entre les concepts présentés précédemment sont décrits dans l'annexe 7.

5.4.4 Liens sémantiques entre les concepts

Une fois les concepts définis et organisés sous des liens taxonomiques, il faut définir les liens sémantiques qui les unissent. L'instanciation des relations entre les concepts est l'expression de liens spécifiques qui peuvent difficilement se généraliser [Fortineau 13a]. La figure 5.5 définit les relations sémantiques entre les concepts de la CMO. Le langage OWL-DL permet deux types de propriétés : des propriétés de type *objet* qui relient deux instances entre elles, et des propriétés de types *données* qui attribuent des valeurs aux instances [Yang 09]. Une propriété possède un *domaine de définition* (domain) et un *domaine d'application* (range).
L'entité centrale du modèle est le **Service** car c'est l'élément qui lie les fournisseurs et les utilisateurs dans la plateforme. Les instances de la classe **Actor** peuvent fournir ou consommer un service. Les instances de la classe **State** permettent de décrire l'état d'un service. La classe **Property** et ses sous-classes définissent les caractéristiques d'un service. Le concept **Deployment_model** désigne les modèles de déploiement du Cloud (public, privé ou hybride). **Semantic_elements** spécifie les méthodes de communication avec le**Service** comme le format de données supporté en entrée et généré à la fin du traitement. Les ressources appartiennent à un **Actor** (Une entreprise par exemple) et sont virtualisées et fournies en tant que **Service**. Un **Service** peut avoir une **PaymentMethod**, **DeliveryMethod** ainsi qu'un ensemble de garanties présentées sous la forme d'un **SLA**. Enfin, un **Service** a un **CustomerFeedback** de la part du client pour évaluer sa qualité.

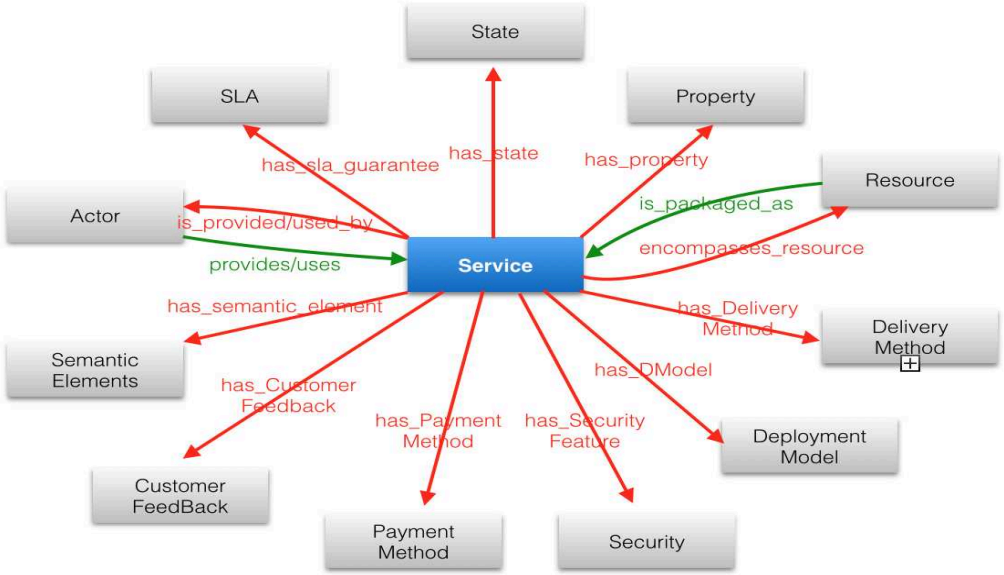


FIGURE 5.5 – *Les lines sémantiques entre les concepts génériques*

En plus de ces liens qui placent le service au centre du système, il existe d'autre relations sémantiques liant le reste des concepts entre eux. Ces relations sont présentées dans la table 5.1.

Property	Domain	Range
has_paymentMethod	Service	PaymentMethod
service_has_property	Service	Property
has_deliveryMethod	Service	DeliveryMethod
service_has_state	Service	State
encompasses_resource	Service	Resource
is_packaged_as	Resource	Service
has_deployment_model	Service	DeploymentModel
has_user_service_link	Actor	Service
has_service_user_link	Service	Actor
owns_resource	Actor	Resource
is_owned_by	Resource	Actor
has_semantic_elements	Service	Semantic_elements
has_sla	Service	SLA
has_customerFeedback	Service	CustomerFeedBack
has_security_features	Service	Security
has_address	Actor	string
has_name	Actor	string
has_value	Price, Initiation_time, Recovery_time, VM_Price, Storage_price	float
software_has_version	Software	string
storage_capacity_value	storage_capacity	nonNegativeInteger

TABLE 5.1 – *Object properties and data properties of the CMO*

5.4.5 L'ontologie CMO comme un modèle d'inference

Comme expliqué dans la section 5.2.3.2, le raisonnement est la capacité à inférer et découvrir de nouvelles informations à partir des concepts, propriétés et axiomes définis dans l'ontologie. Notre ontologie, en plus d'être une base de connaissance, a été développée à l'aide d'axiomes OWL-DL et diversifiée avec des individus afin de mettre en œuvre le mécanisme d'inférence. Les axiomes étant des restrictions appliquées aux classes et aux propriétés. Selon [Fortineau 13c], OWL-DL possède quatre axiomes de base :

— Disjonction des classes : l'intersection des concepts est vide. Par exemple, la classe

Actor est disjointe avec la classe **Service** : un individu ne peut pas être à la fois acteur du système et service ;
— Sous-classe (Subclassof) : **Design_aas** est une sous-classe de **Service** ;
— Sous-propriété (Subproperty) : les propriétés encompasses_SR (SR : Soft Resource) et *encompasses_HR* (HR : Hard Resource) sont des sous-propriétés de la propriété *encompasses_resource* ;
— Équivalence : l'axiome suivant : **Consumer**≡ **Actor** *that_uses* some **Service** définit un consommateur comme étant un acteur du système qui utilise au moins un service.

[Antoniou 04] considèrent le raisonnement comme un support important à la modélisation car il permet de vérifier la cohérence de l'ontologie et de la connaissance, de vérifier les relations entre les concepts et de classifier les instances automatiquement selon leur appartenances aux différentes classes. Selon [Fiorentini 10] il existe quatre types de raisonnement :

— cohérence : vérifier la cohérence de l'ontologie en vérifiant qu'une classe contient des individus ;
— subsomption : déterminer si la description de la classe A est plus générale que la description de B afin de rendre explicites les liens taxonomiques entre les classes ;
— réalisation : trouver tous les concepts auxquels un individus appartient ;
— vérification et récupération d'individu : trouver tous les individus qui appartiennent à une classe.

Afin d'illustrer le mécanisme d'inférence sur notre ontologie, nous avons défini deux classes comme étant :

— **Consumer** ≡ **Actor** and (*use_cm_service* some **Service**)
— **Manufacturing_aas** ≡ **Service** and (*encompasses_resource* some **Manufacturing_resource**)

Ainsi que trois individus :

— User1 *uses_cm_service* Service3
— Service3 *encompasses_software* MgSoftware
— MgSoftware is_a **Manufacturing_software**

La figure ci-après représente l'information inférée. A partir des définitions données ci-dessus, l'ontologie à permis de déduire que User1 est de type **Consumer** et que Service3 est de types : **Manufacturing_aas** et **SaaS**.
L'inférence est obtenue grâce à des moteurs d'inférences, aussi appelés raisonneurs. Pellet, FACT++ et HermiT sont des exemples de raisonneurs implémentés dans l'outil d'édition d'ontologie Protégé.

5.4.6 Règles SWRL

Il existe deux manière pour exprimer et exécuter des règles sur les modèles OWL : (i) en utilisant des axiomes OWL qui permettent de restreindre un concept ou lui associer des instances automatiquement ; ou (ii) par l'écriture d'une règle SWRL [Fortineau 14].

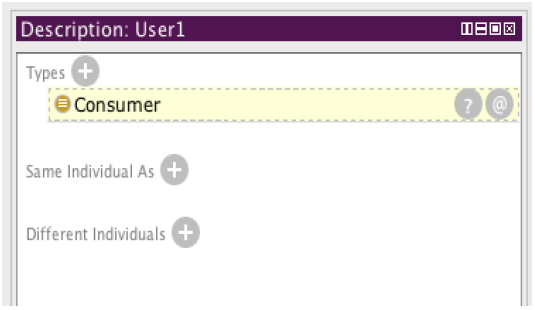


FIGURE 5.6 – *Consumer*

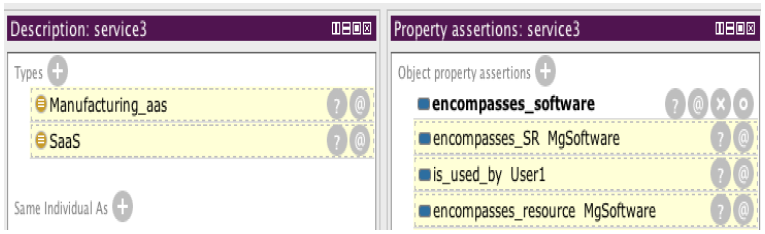


FIGURE 5.7 – *Service 3*

Le Semantic Web Rules Language (SWRL) est un langage qui permet d'intégrer des règles dans l'ontologie pour exprimer des contraintes plus complexes. C'est une combinaison du langange OWL-DL et du Rule Markup Language (RuleML) [Horrocks 04]. L'objectif de SWRL est d'améliorer l'expressivité des langages DL et vise à remédier aux limites des langages ontologiques qui peuvent être résolues en ajoutant des règles à l'ontologie [Eiter 08]. Selon les mêmes auteurs, les règles SWRL sont de type « implication » car composées d'antécédents et de conséquents qui eux mêmes sont composés d'atomes et notés : $antecedent \rightarrow consequent$. Les atomes pouvant être des classes ou propriétés. Une implication SWRL exprime le fait que la conjonction de plusieurs entités permet d'établir de nouvelles propriétés entre les instances.
Dans notre cas, la CMO doit être la plus générique possible, les règles de gestions sont alors spécifiques à chaque entreprise. En effet, ces règles peuvent par exemple représenter des règles d'accès et de visibilité des services pour une entreprise qui choisit de les paramétrer selon les utilisateurs et les droits qu'elle leur attribue. Elle peut restreindre l'accès à ses services si l'utilisateur est un concurrent de cette dernière. Les choix des règles de confidentialité n'est pas toujours le même pour les systèmes du domaine. Par conséquent, ces règles peuvent être modélisées durant l'instanciation sur un système.
Pour illustrer cela, nous avons créé une classe **NotVisible** dans notre ontologie qui contiendra tous les service auxquels un acteur du système User1 n'a pas accès. La règle suivante peut être définie pour aider à classifier les services :

Consumer(User1), is_provided_by(?z, Company-A) $\rightarrow$ *NotVisible(?z)*. Cette règle signifie

que les services fournis par Company-A sont invisibles à l'utilisateur User1 et appartiennent par conséquent à la classe **NotVisible**.

5.5 Validation du modèle proposé

La validation de notre proposition constitue le premier pas vers la mise en place d'une plateforme CM. Nous avons effectué des tests unitaires, présentés dans les sections précédentes afin de valider la CMO. Dans ce qui suit, nous présentons un cas d'utilisation inspiré d'un scénario industriel proposé dans les travaux de [Lu 14] dans le but d'illustrer les processus de classification et de requêtage de services basés sur une matrice de droit grâce à la capacité de raisonnement offerte par les ontologies. Le tableau 5.5 représente la matrice de droit d'accès aux ressources dans le cloud. Par exemple, la première ligne signifie que l'entreprise-A n'a accès qu'à sa propre infrastructure et ses propres services, tous les services proposés par l'entreprise-B, entreprise-C, entreprise-F et seulement à la machine-outil de l'entreprise-F.

Table 5.2 – *Matrice de partage de ressources*

	Entreprise-A	Entreprise-B	Entreprise-C	Entreprise-D	Entreprise-E	Entreprise-F
Entreprise-A	✓	✓	✓	χ	Machine-Outil	✓
Entreprise-B	χ	✓	✓	Machine-Outil	Machine-Outil	χ
Entreprise-C	χ	χ	✓	χ	Machine-Outil	χ
Entreprise-D	χ	✓	SoftResource	✓	MachineTool	χ
Entreprise-E	χ	✓	✓	χ	✓	χ
Entreprise-F	χ	✓	SoftResource	χ	MachineTool	✓

✓ : avoir accès ; χ : *nepasavoiraccs*

Afin de modéliser ces entreprises, leurs ressources et services ainsi que les droits d'accès présentés dans la matrice ci-dessus, nous avons défini six individus de type **Enterprise** : Company-A, Company-B, Company-C, Company-D, Company-E, Company-F. Quatre machine-outils : MachineToolA, MachineToolB , MachineToolD et MachineToolE qui sont des **Manufacturing_equipment** et SoftwareC. Trois individus de type **material** : Iron, Paper et Wood. Trois individus de type **part_size** : P-30, P-50 et P-70.
Nous avons également défini des individus sans spécifier leurs types afin de vérifier si le mécanisme d'inférence les classifie correctement pour valider la CMO. Ces individus sont les suivants : S-A1, S-A2, S-A3, S-A4, S-B1, S-B2, SrvC, SrvD, SrvE, SrvF.
La figure 5.8 présente les taxonomies des classes utilisées dans le scénario.

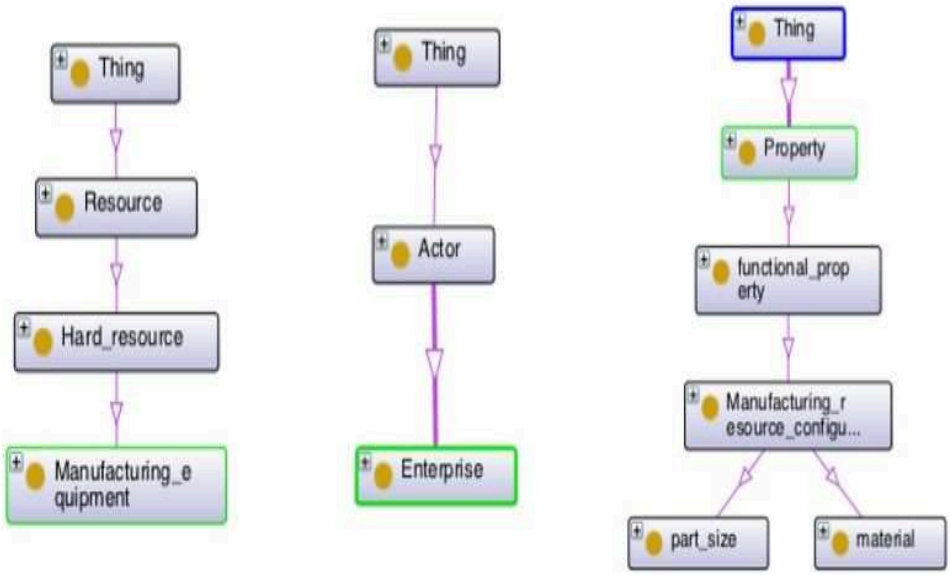


FIGURE 5.8 – *Taxonomies*

La liste des individus ainsi que leurs propriétés est la suivante :

— CompanyA *owns_resource* MachineToolA
— CompanyB *owns_resource* MachineToolB
— CompanyC *owns_resource* SoftwareC
— CompanyD *owns_resource* MachineToolD
— CompanyE *owns_resource* MachineToolE
— CompanyF *provide_cm_service* SrvF
— S-A1 *encompasses_HR* MachineToolA
— S-A1 *has_property* Wood
— S-A1 *has_property* P-30
— S-A2 *encompasses_HR* MachineToolA
— S-A2 *has_property* Wood
— S-A2 *has_property* P-50
— S-A3 *encompasses_HR* MachineToolA
— S-A3 *has_property* Paper
— S-A3 *has_property* P-30
— S-A4 *encompasses_HR* MachineToolA
— S-A4 *has_property* Paper
— S-A4 *has_property* P-50
— SB-1 *encompasses_HR* MachineToolB
— SB-1 *has_property* Paper
— SB-1 *has_property* P-30
— SB-2 *encompasses_HR* MachineToolB
— SB-2 *has_property* Iron
— SB-2 *has_property* P-70
— SoftwareC *is_packaged_as* SrvC
— MachineToolD *is_packaged_as* SrvD

— MachineToolE *is_packaged_as* SrvE
Nous avons défini les classes suivantes en nous basant sur les axiomes OWL :

— **holon_service** ≡ Service and (*encompasses_resource* some Manufacturing_equipment)
— **Manufacturing_aas** ≡ Service and (*encompasses_resource* some (Manufacturing_equipment or Manufacturing_knowledge or Manufacturing_skills or Manufacturing_software))
— **Provider** ≡ Actor and (*provide_cm_service* some Service)

Enfin, nous avons défini six classes **VisibleA, VisibleB, VisibleC, VisibleD, VisibleE, VisibleF** qui contient chacune les services visibles pour une entreprise selon la matrice de droits . La première règle signifie que si un service est fourni par les entreprises suivantes : Company-A, Company-B, Company-C or Company-E alors il est visible pour l'entreprise-A et fait donc partie de la classe **VisibleA**.

— is_provided_by(?c, Company-B), is_provided_by(?d, Company-C), is_provided_by(?e, Company-E), is_provided_by(?x, Company-A) → VisibleA(?c), VisibleA(?d), VisibleA(?e), VisibleA(?x)
— Manufacturing_equipment(?q), is_packaged_as(?q, ?x), is_provided_by(?x, Company-D), is_provided_by(?y, Company-C), is_provided_by(?z, Company-B) → VisibleB(?x), VisibleB(?y), VisibleB(?z)
— is_provided_by(?x, Company-C), is_provided_by(?z, Company-E) → VisibleC(?x), VisibleC(?z)
— Soft_resource(?w), is_owned_by(?w, Company-C), is_packaged_as(?w, ?l), is_provided_by(?m, Company-B), is_provided_by(?n, Company-B), is_provided_by(?o, Company-D), is_provided_by(?p, Company-E) → VisibleD(?l), VisibleD(?m), VisibleD(?n), VisibleD(?o), VisibleD(?p)
— is_provided_by(?a, Company-B), is_provided_by(?b, Company-C), is_provided_by(?c, Company-E) → VisibleE(?a), VisibleE(?b), VisibleE(?c)
— Soft_resource(?p), is_owned_by(?p, Company-C), is_packaged_as(?p, ?m), is_provided_by(?m, Company-C), is_provided_by(?x, Company-B), is_provided_by(?y, Company-E), is_provided_by(?z, Company-F) → VisibleF(?m), VisibleF(?x), VisibleF(?y), VisibleF(?z)

L'ontologie a inféré de nouvelles informations déduites à partir des axiomes et règles définis ci-dessus. Par exemple, le raisonneur a classé **holon_service** ⊆ **Manufacturing_aas**. Quant aux individus, le raisonneur a fait les déductions suivantes :

— S-A1∈ **holon_service**
— S-A1 *encomposses_resource* machineTool-A
— S-A1 *is_provided_by* Company-A
— machineTool-A *is_packaged_as* S-A1
— machineTool-A *is_owned_by* Company-A
— Wood *is_property_of_service* S-A1
— Company-A ∈ **Provider**
— Company-A *provide_cm_service* S-A1

Les classes notées **VisibleX** contiennent les service auxquels une Entreprise-X a accès.

Par exemple l'entreprise-A peut accéder aux services suivants : S-A1, S-A2, S-A3, S-A4, SB-1, SB-2, SrvC, SrvE.

— **VisibleA** : S-A1, S-A2, S-A3, S-A4, SB-1, SB-2, SrvC, SrvE
— **VisibleB** : SB-1, SB-2, SrvC, SrvD
— **VisibleC** : SrvC, ServE
— **VisibleD** : SB-1, SB-2, SrvC, SrvD, SrvE
— **VisibleE** : SB-1, SB-2, SrvC, SrvE
— **VisibleF** : SB-1, SB-2, SrvC, SrvE, SrvF

Enfin, nous avons effectué des requêtes DL dans le but de valider le processus d'extraction des informations à partir du contenu de l'ontologie. Ainsi pour la requête **Service** that ((*service_has_property* value paper) and (*service_has_property* value P-30)) l'ontologie a retourné comme réponse : SA-3 and S-B1.

5.6 Conclusion

Nous avons proposé dans ce chapitre la mise en œuvre de la méthodologie ASCI-Onto qui représente notre contribution méthodologique. Nous avons présenté un modèle générique de connaissance basé sur les ontologies. L'ontologie du domaine appelée *Cloud Manufacturing Ontology* (CMO) a été développée en suivant la méthodologie METHONTOLOGY. Les étapes constituant la méthodologie METHONTOLOGY sont les suivantes : spécification, acquisition de connaissance, conceptualisation, intégration, implantation, évaluation et documentation.
Par ailleurs, parmi les différents langages de représentation d'ontologie existants, nous avons choisi le langage OWL. Les étapes de METHONTOLOGY nous ont permis de produire notre ontologie dont les concepts génériques sont les suivants : **Actor, Resource, Service, PaymentMethod, DeploymentModel, Property, CustomerFeedBack, Security, State, DeliveryMethod, SLA, Semantic_elements**. L'ontologie regroupe des concepts liés au CM et également des concepts liés à l'infrastructure locale. Les liens sémantiques entre ces concepts placent le **Service** au centre de l'ontologie. La CMO constitue à la fois une base de connaissance, et un modèle d'inférence où des axiomes et des règles SWRL ont été définies afin de déduire de nouvelles connaissances.
Notre formalisme a été validé sur la base d'un scénario extrait de la littérature scientifique. La définition de classes et de règles SWRL nous a permis d'inférer de nouvelles informations et l'ensemble de l'ontologie à permis de donner les résultats escomptés aux requêtes DL.

6

Proposition méthodologique pour la construction d'une architecture CM

Résumé du chapitre

Nous avons présenté dans le chapitre précédent le modèle ontologique permettant de modéliser une plate-forme CM. Dans ce chapitre, nous présentons la deuxième partie de notre méthodologie : ASCI-Onto qui est la conception et l'implémentation d'une bibliothèque de composants logiciels. Les étapes mises en place afin de réaliser cela sont les suivantes :

- *expliquer la relation entre le modèle sémantique (CMO et l'architecture finale) ;*
- *dériver l'ontologie CMO en diagramme UML afin de réaliser l'étape de conception de la plate-forme CM ;*
- *présenter des simulateurs de plate-forme CC et expliquer notre choix pour CloudSim et comment ce dernier sera modifié dans le but de réaliser une simulation CM ;*
- *enfin, présenter l'architecture envisagée, le choix du langage d'implémentation et un exemple de cas d'étude illustrant le fonctionnement de la plate-forme.*

Sommaire

6.1 Relation entre le modèle sémantique et l'architecture CM

Nous avons présenté dans le chapitre précédent notre modèle générique de connaissance, la CMO qui a été développée en utilisant les ontologies. Ce modèle constitue la première étape de notre méthodologie de mise en place d'une plate-forme CM basée sur l'approche ASCI-Onto. Il est question dans ce chapitre d'utiliser ce modèle pour compléter les deux étapes de notre méthodologie, c'est-à-dire : la Conception et l'Implémentation afin d'élaborer notre bibliothèque de composants du domaine. La figure 6.1 présente le positionnement du chapitre 6 dans notre méthodologie. Nous utilisons ensuite la bibliothèque de composants créée pour illustrer le fonctionnement d'une plate-forme CM sur un scénario.

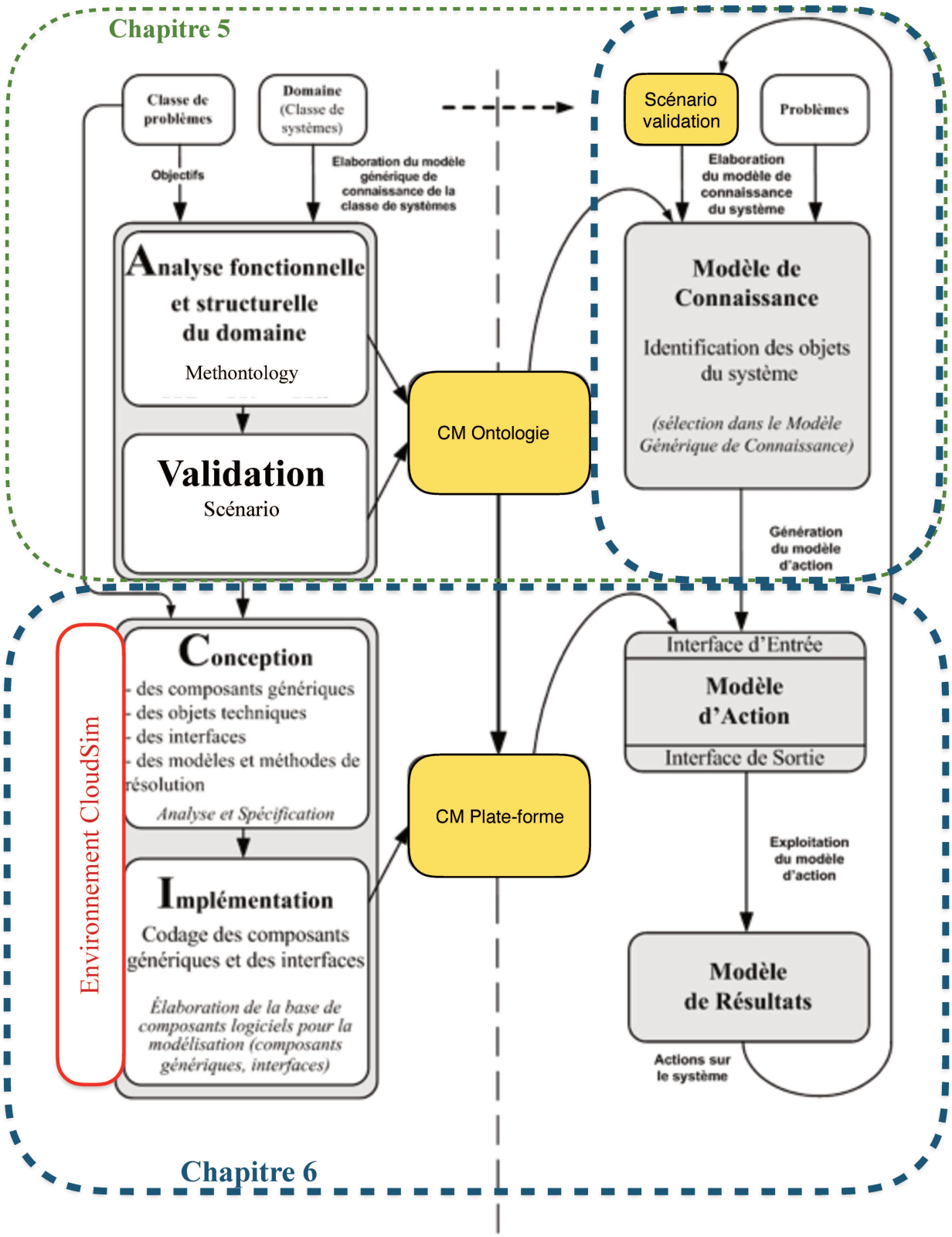


FIGURE 6.1 – *Positionnement du chapitre 6 dans la thèse*

La mise en place de la bibliothèque de composants est réalisée grâce au modèle générique de connaissance. Ainsi, la base de connaissance est retranscrite sous forme de composants informatiques pour valider la méthodologie et mettre en pratique le fonctionnement d'un système CM. A ce stade, deux schémas sont mis en relation, le schéma sémantique et le schéma technique. La figure 6.2 illustre le rôle de la CMO et son lien avec la plate-forme CM. Le schéma sémantique représente les correspondances (mappings) entre l'ontologie de domaine et les descriptions des fournisseurs réalisées grâce au partage d'une sémantique commune. Le schéma technique décrit l'utilisation de l'ontologie de domaine par la plate-forme CM. Le schéma techniques est basé sur la définition proposée par [Xu 12b] (section 3.4). Le médiateur représente le *Global Service Layer* (GSL) car il constitue la couche intermédiaire entre l'utilisateur (*User Domain* (UD)) et les ressources (*Manufacturing Resource Layer* (MRL)).

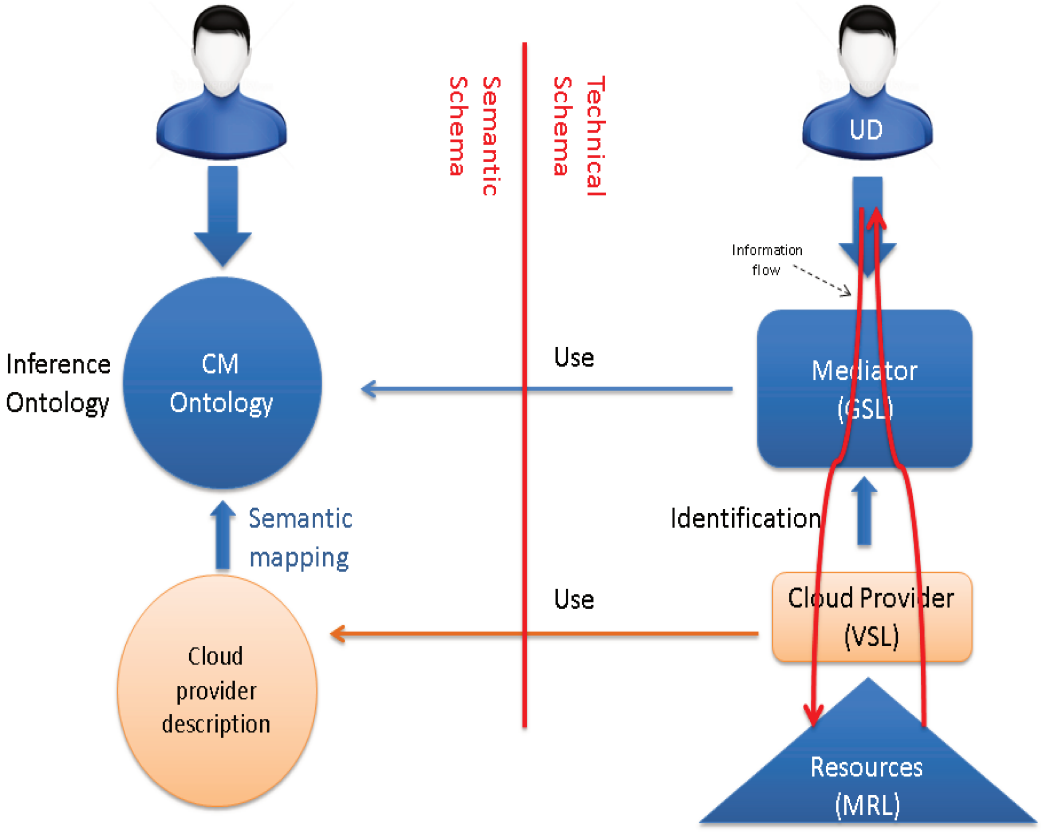


FIGURE 6.2 – *Lien entre schéma sémantique et schéma technique*

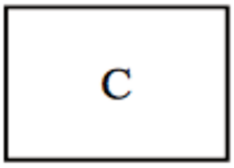
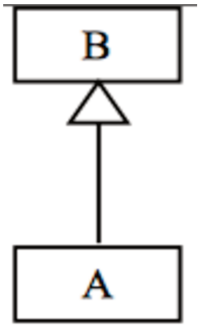
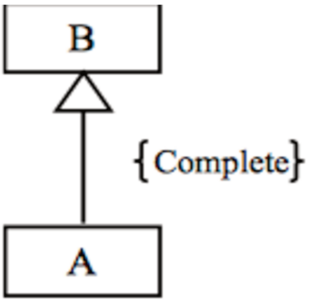
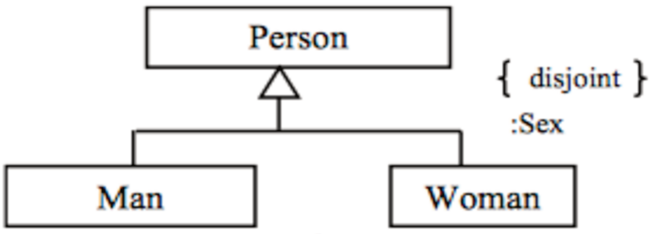
6.2 La conception de la bibliothèque de composants logiciels

La méthodologie ASCI préconise la conception d'une bibliothèque de composants logiciels génériques pouvant être utilisés pour la réalisation des modèles d'action des systèmes du domaine étudié. La conception des composants logiciels consiste à formaliser un ensemble de composants sans prendre en compte les détails de l'implémentation tels que les langages. Pour pouvoir concevoir les composants logiciels, une étape préliminaire de dérivation de l'ontologie en un diagramme UML doit être réa-

lisée. D'un point de vue technique, les outils d'implémentation d'ontologies font face à des lacunes de robustesse et de maturité. [Fortineau 13a] explique que les difficultés d'implémentation des ontologies ne sont pas seulement liées à un manque d'efforts de développement par rapport aux technologies UML mais que OWL possède pas moins de quatre syntaxes et c'est cette multiplicité syntaxique qui freine le développement d'outils, limité également par la performance des moteurs d'inférence.
D'autre part, l'utilisation du langage UML permet de :
— obtenir une modélisation de très haut niveau indépendante des langages et des environnements ;
— faire collaborer des participants de tous les horizons autour d'un même document de synthèse ;
— faire des simulations avant de construire un système ;
— exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications, etc.
Par conséquent, la transformation de notre ontologie en diagramme UML est une étape intermédiaire et nécessaire pour l'implémentation d'une plate-forme CM. Nous présentons et illustrons le concept de dérivation dans la sous-section suivante.

6.2.1 La dérivation de l'ontologie CMO en diagramme UML

La dérivation dans notre cas signifie le passage de l'ontologie représentée dans le langage OWL au diagramme de classes. Le diagramme de classe est un diagramme UML considéré comme un diagramme de classification. L'objectif de ce diagramme est de montrer la structure statique d'un système. Il expose les types et les instances du système ainsi que les relations entre eux. Ce type de diagramme peut être utilisé par les architectes systèmes pour vérifier et tester leur conception et par les développeurs pour concevoir et documenter le système codé.
Afin de réaliser la dérivation de l'ontologie CMO, nous nous basons sur les travaux de [Kiko 06] où il présente les correspondances entre OWL et UML que nous résumons dans le tableau ci-après. Nous présentons dans ce tableau les notions OWL qui possèdent un équivalent en UML et inversement.

OWL	UML
Classe C	
A subClassOf B	 Utilisation d'une relation de généralisation
A equivalentClass B	 Utilisation d'une relation de généralisation de type « ensemble complet»
A disjointWith B	 Utilisation d'une relation de généralisation de type « ensemble disjoint » basé sur le critère « Sexe »

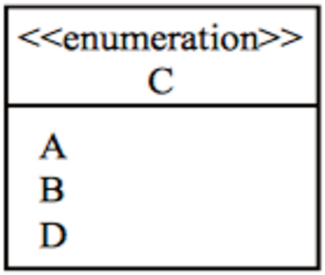
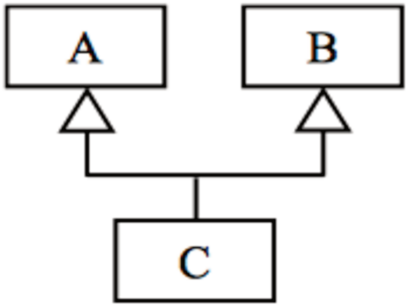
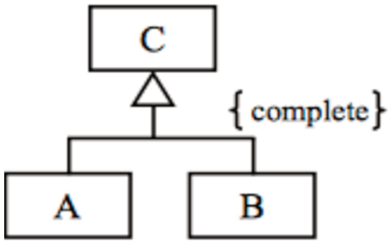
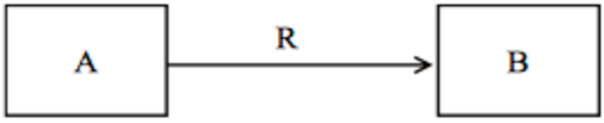
OWL	UML
Enumération : C oneOf A, B, D	 Utilisation d'une classe de type « enumeration » qui contient une liste des individus
C subClassOf intersectionOf A and B	 Utilisation de la notion de l'héritage multiple
C unionOf A and B	 Utilisation d'une relation de généralisation de type « ensemble de sous-classe »
R domain A Range B	 Utilisation d'une relation d'association d'une classe A vers une classe B

FIGURE 6.3 – *Lien entre schéma sémantique et schéma technique [Kiko 06]*

Nous présentons dans la figure 6.4 un modèle UML dérivé à partir de la CMO. Afin de ne pas charger la figure ce schéma est partiel et ne montre pas tous les détails des classes représentées. Cette transformation est basée sur le tableau présenté ci-dessus avec quelques exceptions. Par exemple dans la cas de le visibilité d'un service selon le modèle de déploiement utilisé que nous avons choisi de définir comme attribut dans le classe Service. En effet, selon [Kiko 06] une classe **Service** doit être liée à une classe **DeploymentModel** par un lien d'association et que cette dernière est de type « énumération » qui contient les termes {public, privé, hybride, communautaire}.
Il est important de souligner que cette transformation engendre une perte d'expressivité et de capacité d'inférence offertes par le langage OWL. D'autre part, l'expression des règles SWRL permettant la découverte de nouvelles informations n'est pas autorisé dans UML. Par conséquent, nous avons utilisé UML pour représenter la partie « statique » de la CMO c'est à dire la CMO en tant que base de connaissance (section 5.4.3). La capacité d'inférence, perdue durant la transformation a été implémentée durant la phase *Implémentation* de la méthodologie ASCI-Onto.

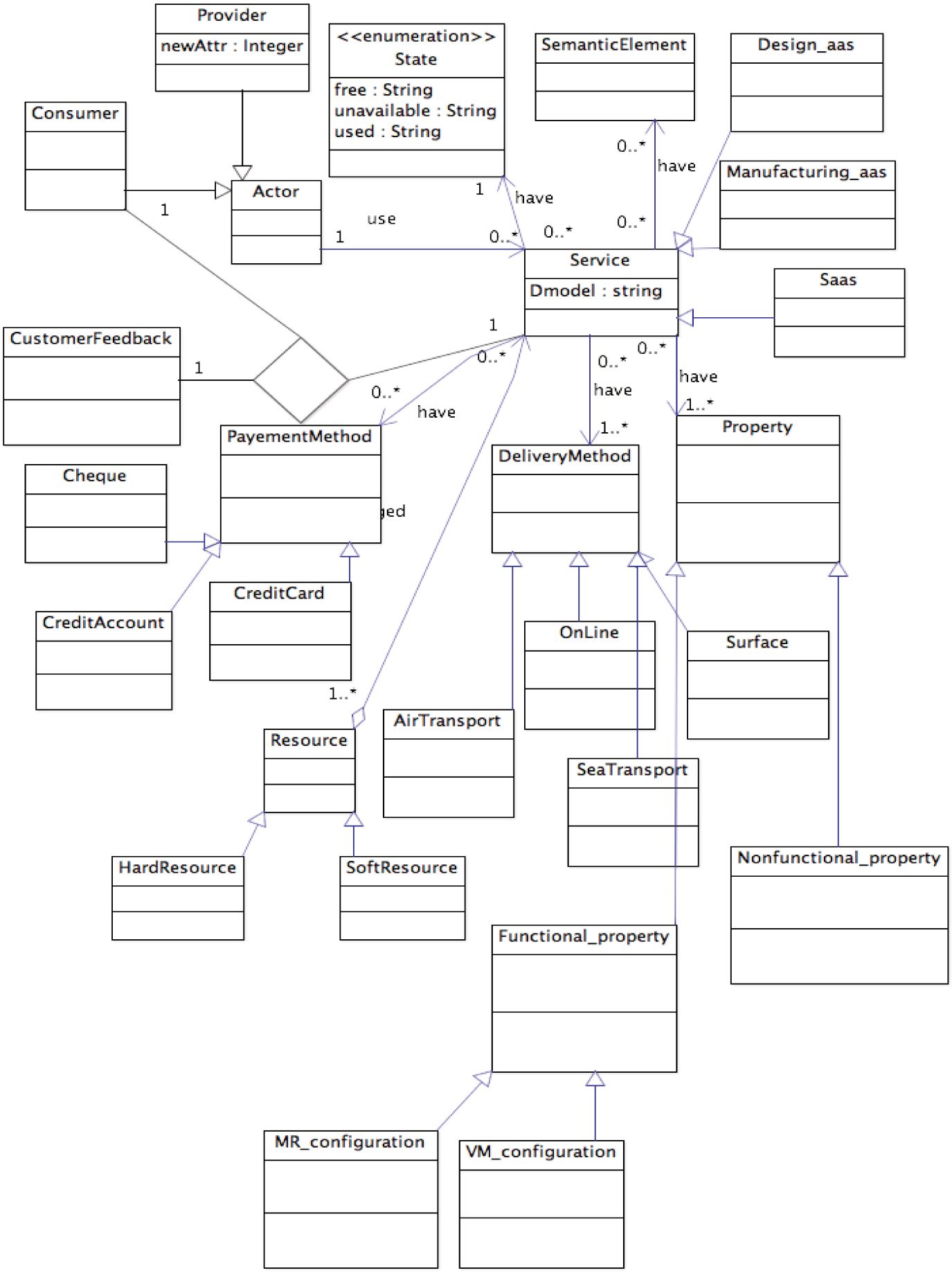


FIGURE 6.4 – *Diagramme de classe dérivé à partir de la CMO*

En suivant le tableau d'équivalence entre OWL et UML, nous avons dérivé des classes en attributs afin de définir par exemple les caractéristiques des ressources manufacturières et des machines virtuelles, les informations concernant les services et les acteurs du systèmes, etc. Ainsi, certaines classes ont disparu du diagramme UML pour être incluses en tant qu'attributs dans d'autres classes.
Les méthodes des classes servent, outre des méthodes de création d'instances d'initialisation de ressources par exemple et de services, à manipuler les données relatives à ces instances et à assurer la communication entre celles-ci. Ces méthodes seront abordées en détails dans la partie implémentation.

L'étape de conception ne s'arrête pas à la dérivation du diagramme de classe, car ce diagramme sera fusionné au diagramme de classe représentant l'outil de simulation utilisé par la suite pour la validation de notre architecture. La section suivante présente un aperçu des outils de simulation existants ainsi que « CloudSim » ; l'outil que nous avons choisi d'utiliser.

6.2.2 Pourquoi utiliser la simulation

Un modèle de simulation est le nom donnée au processus d'utilisation de formules, règles et environnements pour effectuer des expérimentations et produire un modèle d'un système physique (biologie, météorologie, etc), un système humain (économie, géographie, etc) ou une conception d'un produit comme dans le domaine de l'ingénierie (avions, voitures, etc) à des fins de tests. Dans le domaine du cloud la simulation permet l'expérimentation à faible coût dans des environnements à faible risque. En effet, [Calheiros 11] expliquent que l'utilisation d'infrastructure réelle, comme Amazon EC2 et Microsoft Azure pour tester les performances des applications dans des conditions variables (la disponibilité, la charge de travail, etc) est souvent limitée par la rigidité de cette infrastructure. De plus, configurer et paramétrer des infrastructures à grande échelle afin d'effectuer des tests représentent un travail fastidieux. Par conséquent, il n'est pas possible de réaliser des expériences de manière répétitive et fiable dans des environnement Cloud réels. Les approches basées sur la simulation offrent aux entreprises des avantages importants en leur permettant de [Calheiros 11] : (i) tester leurs services dans un environnement reproductible et contrôlable ; (ii) identifier le goulot d'étranglement d'un système avant de déployer les solutions et (iii) expérimenter sur des ressources avec des différentes charges de travail et de performances afin de tester le comportement du système face aux demande d'adaptation et passage à l'échelle.

6.2.3 Quelques outils de simulations d'environnements CC

Récemment, de nombreux outils de simulation Cloud on été proposés afin d'offrir des environnement permettant une évaluation reproductible et contrôlée des méthodes de gestion des ressources et applications dans le cloud .
GreenCloud [Kliazovich 10] est un simulateur de cloud sur la recherche de l'efficacité énergétique. C'est une extension du simulateur NS2 [1] et est capable de calculer la consommation d'énergie des ressources.
MDCsim [Lim 09] a été développé à *the Pennsylvania State University* en 2009. Il est

1. http://www.isi.edu/nsnam/ns/ consulté en Décembre 2015.

fourni avec des caractéristiques matérielles spécifiques provenant de différents fournisseurs. MDCsim permet d'estimer la consommation d'énergie et est un produit commercial qui n'est pas disponible pour un téléchargement public.
CloudSim est probablement le plus sophistiqué des simulateurs cloud [Kliazovich 10]. Il a été développé au laboratoire *CLOUDS Laboratory* dans le département « Science and Software Engineering » de l'université de Melbourne, en Australie en 2002. C'est un simulateur à événements discrets implémenté en Java, permettant la simulation des centres de données (Data Centers), de leurs hotes, ressources et machines virtuelles. Afin de réaliser un modèle de simulation CM nous avons choisi d'utiliser CloudSim car il est le plus flexible et le mieux adapté pour la simulation d'environnements CC. Le modèle et l'architecture de cet outil sont présentés dans la section suivante.

6.2.4 CloudSim ToolKit

CloudSim est un framework, développé en Java, qui modélise et simule l'environnement du CC [Calheiros 11].
La figure 6.5 illustre les différentes couches de la structure de CloudSim et ses éléments architecturaux. Au niveau le plus bas se trouve le moteur de simulation à événements discrets SimJava, qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur ; telles que les files d'attente, le traitement des événements, création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation.

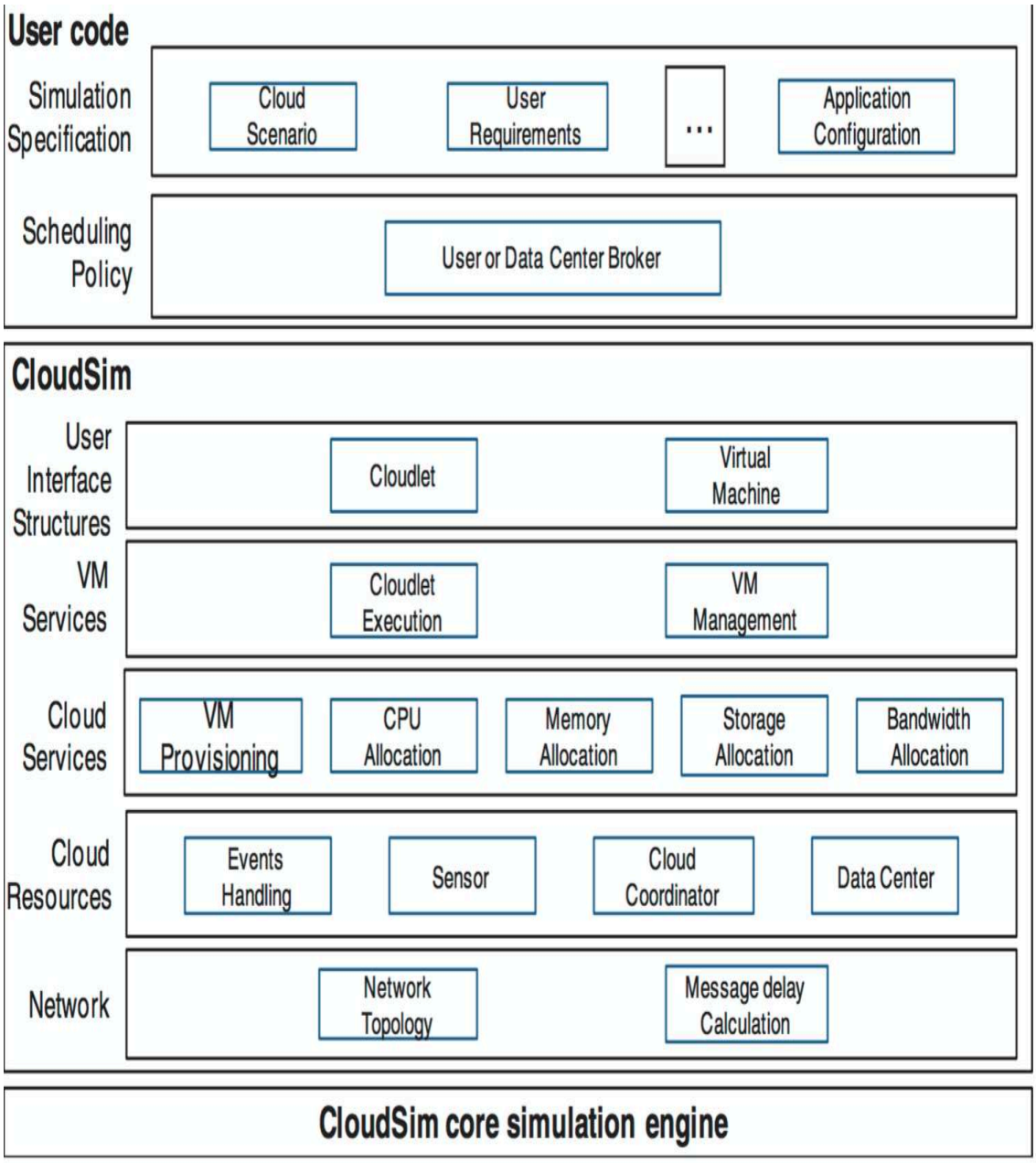


FIGURE 6.5 – *L'architecture de CloudSim [Calheiros 11]*

La couche *User code* contient les informations de base nécessaires au bon fonctionnement des hôtes et des applications comme le nombre de machines, leurs spécifications, le choix des stratégies d'ordonnancement du broker, etc. La couche *CloudSim* fournit un support pour la modélisation des environnements data centers virtualisés avec des classes permettant la gestion et la configuration des machines virtuelles, mémoire, stockage, etc. Dans cette couche, sont aussi gérées les stratégies d'allocation de machines virtuelles et les topologies des différents réseaux. Les classes principales implémentées dans cette couches sont les suivantes (pour plus d'informations sur toutes les classes vous pouvez consulter l'api CloudSim 3.0) :

— **Datacenter** est composé d'un ensemble de hôtes et est responsable de la gestion

des machines virtuelles. Son comportement est similaire à un fournisseur Iaas car il reçoit des requêtes de la part du broker (définition 1.8) pour créer les machines virtuelles à l'intérieur des hôtes.

— **DatacenterBroker** représente le courtier agissant pour le compte d'un utilisateur. Il assure deux mécanismes dans le simulateur : celui de la demande d'allocation de machines virtuelles dans les data centers ainsi que celui de la soumission de tâches à exécuter par les machines virtuelles. Cette classe peut être étendues par les utilisateurs de CloudSim afin de mener leur expériences en utilisant leur propres politiques.

— **Host** exécute des actions liées à la gestion des machines virtuelles telles que la création et la destruction de ces dernières. Une machine hôte possède sa propre politique d'allocation de mémoire, d'éléments de traitement (processeur) et de bande passante au machines virtuelles qu'elle héberge. Une machine hôte est associée à un data center.

— **VM** représente une entité logique qui fonctionne comme une machine physique et exécute les tâches reçues par les brokers.

— **Cloudlet** est une classe qui représente une tâche. Elle contient l'identifiant de l'utilisateur ainsi que toutes les informations sur les tâches à exécuter sur les machines virtuelles.

6.3 CloudSim adapté aux CM

Les liens entre les classes principales de CloudSim sont représentés dans la figure 6.6.

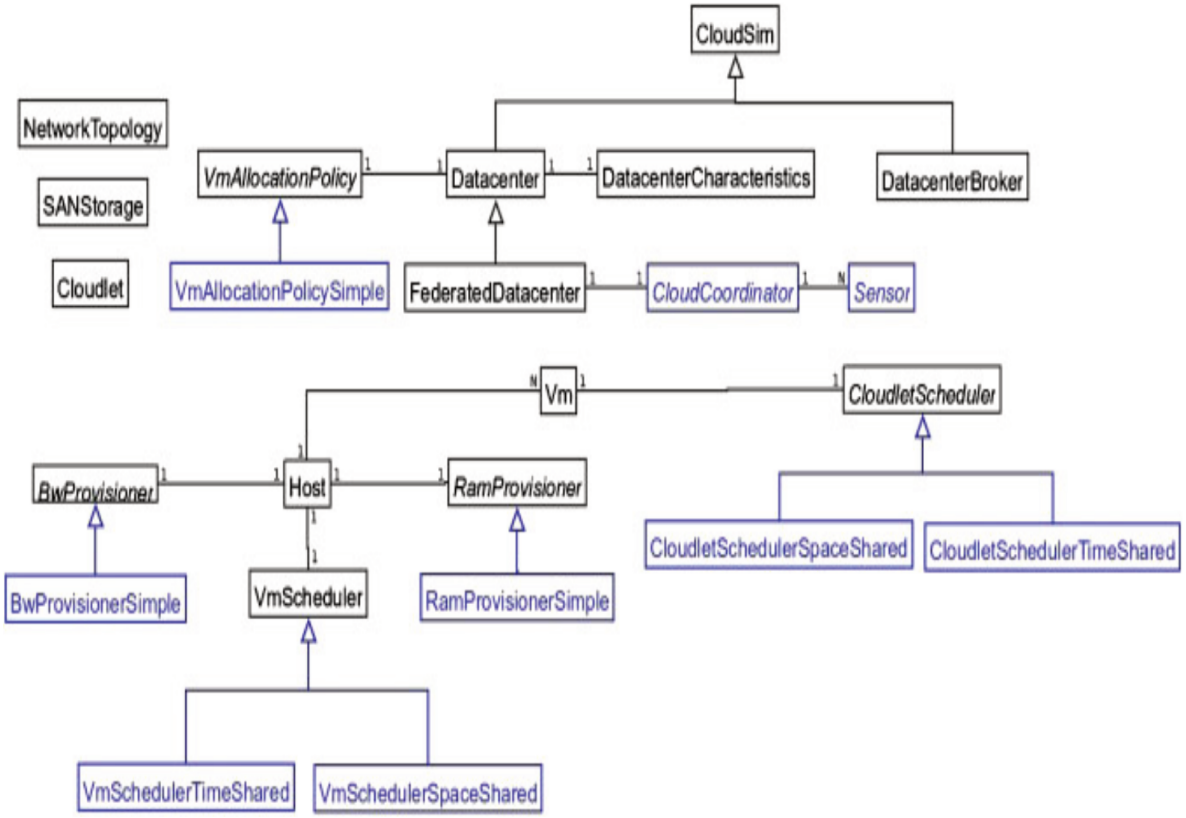


FIGURE 6.6 – *Le diagramme de classe de CloudSim [Calheiros 11]*

Le diagramme UML est défini en plusieurs couches : la première couche *CloudSim* représente la classe génerle qui permet de simuler un environnement CC dans lequel toutes les entités seront modélisées. La deuxième couche englobe *Datacenter* et *DatacenterBroker* qui représente l'abstraction faite de ressources physiques et leurs gestionnaires. Les serveurs *hotes* sont définis dans le troisième niveau, et sont encapsulés dans des *Datacenters*. Un *Datacenter* peut supporter plusieurs *hotes* et plusieurs machines virtuelles (*Virtual machine* (VM)) peuvent être créées dans une *hote*. Trois autres classes sont définies séparément : *NetworkTopology, SANStorage* et *Cloudlet*.

Les classes *DatacenterCharacteristics* et *VmAllocationPolicy* sont liées par une association de cardinalité 1 à la classe *Datacenter*. *DatacenterCharacteristics* décrit les caractéristiques d'un *Datacenter*, comme l'architecture du système, le système d'opération, le mécanisme de tarification, etc. *VmAllocationPolicy* est une classe abstraite qui représente la politique d'approvisionnement des hôtes et VM.
Par défaut, CloudSim met en œuvre une politique simple qui crée des machines virtuelles sur le hôte disponible (premier arrivé, premier servi). Cette politique est définie par la classe *VmAllocationPolicySimple*. La classe *FederatedDatacenter* permet de modéliser des Clouds fédérés ou inter-connectés. L'entité *CloudCoordinator* est nécessaire fin de réaliser la fédération de plusieurs Clouds. Cette classe est responsable de la communication avec d'autres Datacenters et utilisateurs dans l'environnement de simulation. L'autre fonctionnalité importante de *CloudCoordinator* est le monitoring, qui est mis en

œuvre par la définition de la classe *Sensor*. La classe *Sensor* représente des événements spécifiques qui peuvent influencer le processus d'approvisionnement des ressources, le non respect du SLA, etc. La classe *Host*(Hote) représente une ressource physique comme par exemple un serveur. Elle encapsule des informations importantes sur les ressources comme par exemple la mémoire, capacité de stockage, etc.
Les classes *BwProvisioner, RamProvisioner* et *VmScheduler* sont toutes des classes abstraites qui fournissent des composants pour définir des politiques de gestion de bande passante, la gestion de la RAM et de la gestion des VM pour un serveur.
Les *Cloudlets* qui représentent les tâches à exécuter sur les VM, sont gérés par la classe *CloudletScheduler.* sur la base des deux politiques définies : temps partagé ou espace partagée.
NetworkTopology définit la topologie d'un environnement de CC, *SANStorage* représente un réseau de stockage constituée d'un ensemble de disques durs connectés à un réseau local. La capacité globale du SAN est définie à partir des capacités des disques connectés.
Deux autres classes essentielles pour la simulation de CC mais qui ne sont pas représentés dans la figure 6.6 :
CloudInformationService : un CIS est une entité qui assure l'enregistrement des ressources et offre des capacités de découverte de service.
SimEntity : une classe abstraite, qui représente une entité de simulation qui est capable d'envoyer des messages à d'autres entités, de traiter les messages reçus et gérer les événements. *Datacenter, DatacenterBroker, Host* et *VM* sont toutes des sous-classes de *SimEntity*.
Nous remarquons qu'il existe dans CloudSim des classes qui gèrent la partie CC et que notre contribution consiste à intégrer la partie CM afin de créer un environnement complet pour la simulation d'une plate-forme CM. D'autre part, certaines classes comme *Cloudlet*, *DatacenterBroker*, *CloudletScheduler* peuvent être modifiées pour prendre en compte la partie CM. La figure 6.7 montre les classes que nous ajoutons à l'environnement CloudSim. Ces classes ainsi que leurs sous-classe sont implémentées afin de permettre la simulation d'un environnement CM.

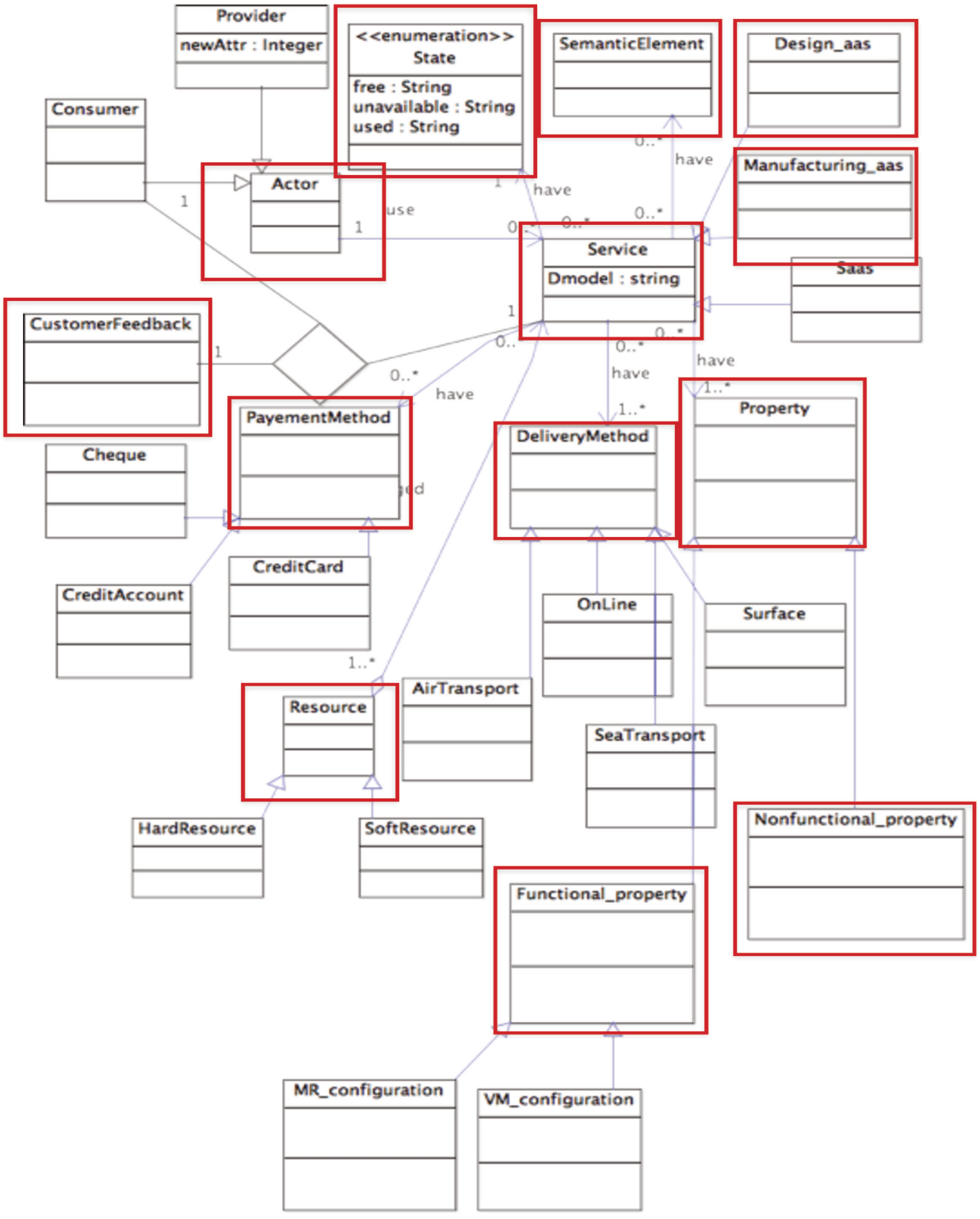


FIGURE 6.7 – *Les classes qui seront integrées dans l'envrionnement CloudSim*

6.4 Implémentation

La méthodologie ASCI conduit à la conception d'un environnement de modélisation et le développement de l'environnement logiciel qui s'y rapporte. La conception et l'implémentation sont définies comme étant un ensemble comprenant [Rodier 10] (figure 6.8) :

1. **Un logiciel d'Évaluation de Performances** (noyau de l'environnement) qui est, en général, un modèle de simulation à événement discrets.
2. **Une couche Graphique** qui est composée d'outils autorisant une exploitation conviviale de l'environnement.
3. **Une couche Aide à la Décision** qui doit permettre à l'expert d'accéder aisément à des outils d'analyse des résultats.
4. **Une couche Gestion de Données** qui concerne l'accès à des bases de données en vue, par exemple, de recueillir les caractéristiques des entités, ...
5. **Une couche Recherche Opérationnelle et Statistique** qui permet d'exploiter facilement ces techniques pour aider à l'interprétation des résultats.
6. **Une couche Méthodes d'Analyse et de Spécification et Outils de Spécification** qui contient des méthodes et des outils d'analyse et de spécification pour décrire, d'une part, la structure du système, et d'autre part, pour spécifier les flux et les règles de gestion du système.

Ces couches communiquent à l'aide d'**Interfaces**.

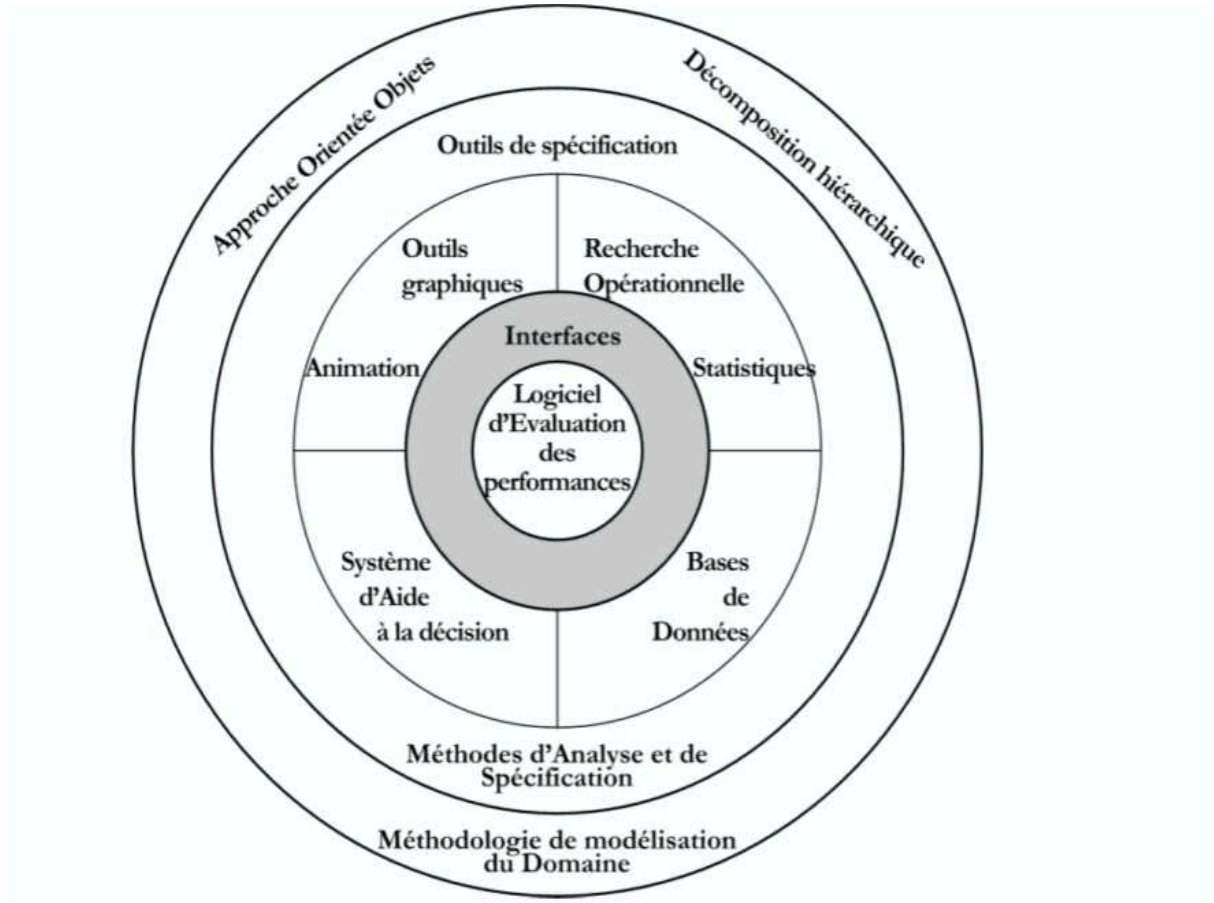


FIGURE 6.8 – *Définition d'un environnement de modélisation [Rodier 10]*

Dans ce qui suit nous présentons les langages et outils qui nous ont permis de mettre en place l'architecture CM en nous basant sur les étapes préconisées par ASCI.

6.4.1 Choix du langages et logiciel de programmation

Pour la mise en place de l'architecture proposée et la réalisation de la simulation d'une plateforme CM, nous avons choisi d'utiliser le langage Java pour les raisons suivantes :

— Java est orienté objet, cela permet de créer des programmes modulaires qui contiennent un code réutilisable ;
— Java est indépendant de toute plateforme ce qui facilite la « portabilité » des programmes depuis un système d'exploitation vers un autre. Cette propriétés est primordiale pour le développement d'application et logiciels orientés Web ;
— Java considère la sécurité comme un besoin important ainsi le compilateur java, l'interpréteur et l'environnement d'exécution ont été développés en prenant en compte la sécurité ;
— C'est un langage multithread [2] permettant l'exécution rapide des programmes ;
— Il est usuellement utilisé dans l'intégration d'entreprise ;
— Il existe de nombreuses bibliothèque disponibles, de qualité industrielle, libres et open source ;
— le simulateur CloudSim est implémenté en Java.

2. Un langage multithread est capable d'exécuter plusieurs tâches simultanément.

Nous utilisons l'environnement Eclipse [3] pour l'implémentation de notre architecture avec le simulateur CloudSim.

6.4.2 Architecture envisagée

Nous présentons dans cette section l'architecture envisagée qui permet d'implémenter le modèle UML dérivé à partir de la CMO et l'integrer à l'environnement de simulation CloudSim. Cette intégration permet ainsi de faire évoluer l'outil CloudSim, qui est initialement développé pour effectuer des simulations CC, vers un outil de simulation de plate-forme CM. Cette architecture est représentée par la figure 6.9.

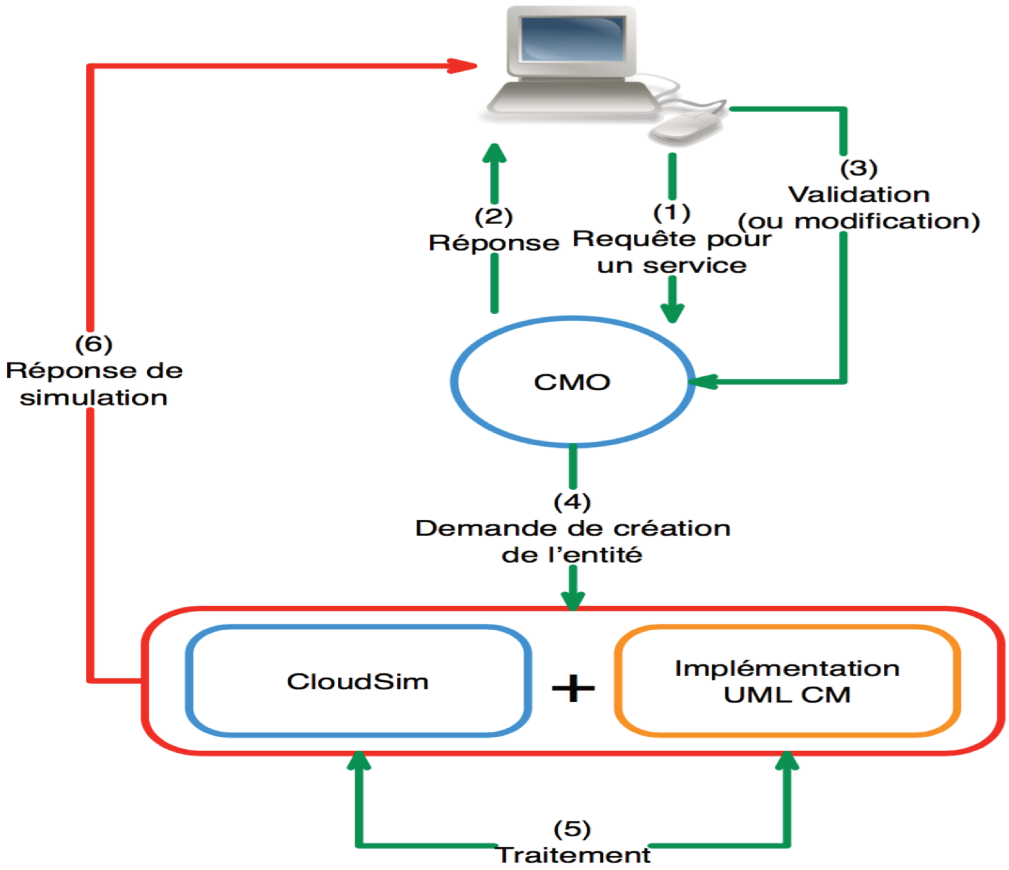


FIGURE 6.9 – *L'architecture envisagée*

La CMO est au cœur de l'architecture car elle constitue l'élément majeur de la mise en correspondance des utilisateurs et fournisseurs. La première étape du fonctionnement générale de la plate-forme est la publication de services existants. Le processus de communication commence par l'action effectué par les fournisseurs qui consiste à s'enregistrer et publier les services qu'ils proposent. Le processus de traitement de la requête client est constitué d'un ensemble d'étapes :

1. l'utilisateur envoie sa requête sur la plate-forme, au module représentant la base de connaissance contenant toutes informations concernant les services disponibles qui n'est autre que l'implémentation de la CMO en tant que base de connaissance ;

3. https://eclipse.org/downloads/ consulté en Décembre 2015

2. la plate-forme renvoie à l'utilisateur la réponse à sa requête avec une liste de services appropriés ;
3. à cette étape, l'utilisateur peut choisir un service parmi ceux proposés ou décider de modifier sa requête ;
4. après confirmation de l'utilisateur, une demande de création de l'entité (machine virtuelle, simulation d'une machine-outil, etc) est envoyée à la plate-forme ;
5. l'entité est ensuite créée et les résultats de simulations sont envoyés à l'utilisateur.

Le module représentant la CMO (figure 6.10) est en réalité divisé en deux parties : la première partie : la base de connaissance dont la structure a été dérivée en diagramme UML permet de représenter notre domaine. Les instances du domaines (services, fournisseurs, utilisateurs, etc) sont représentées dans un fichier XML. La deuxième partie : le raisonneur qui est une implémentation de la partie raisonnement de la CMO et qui agit comme un moteur de recherche qui lie les informations saisies par l'utilisateur, parse la base de connaissance pour trouver les services adéquats et envoie une requête au simulateur pour créer les entités demandées. En effet, nous avons expliqué dans la section 6.2, les outils d'implémentation d'ontologie manquent aujourd'hui de robustesse et de maturité et rencontrent des lacunes quant à l'implémentation de la partie « raisonnement » des ontologies.

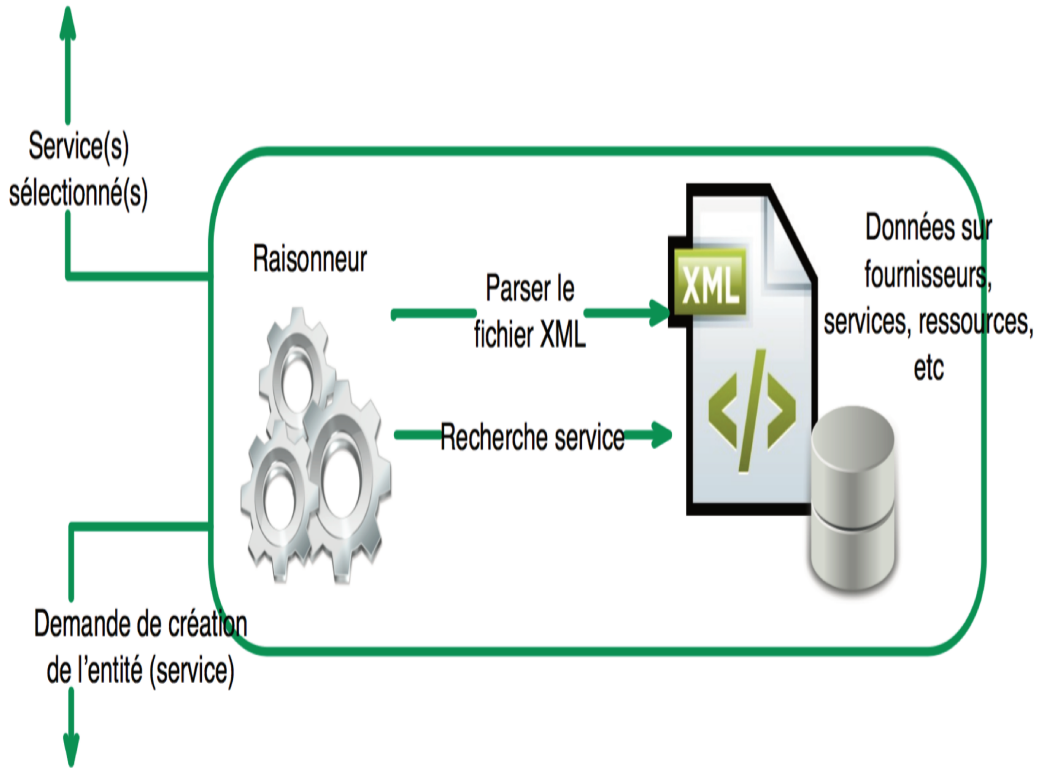


FIGURE 6.10 – *Détails du module CMO*

La figure 6.11 donne l'architecture de l'environnement logiciel en instanciant la figure 6.8 à notre proposition architecturale.

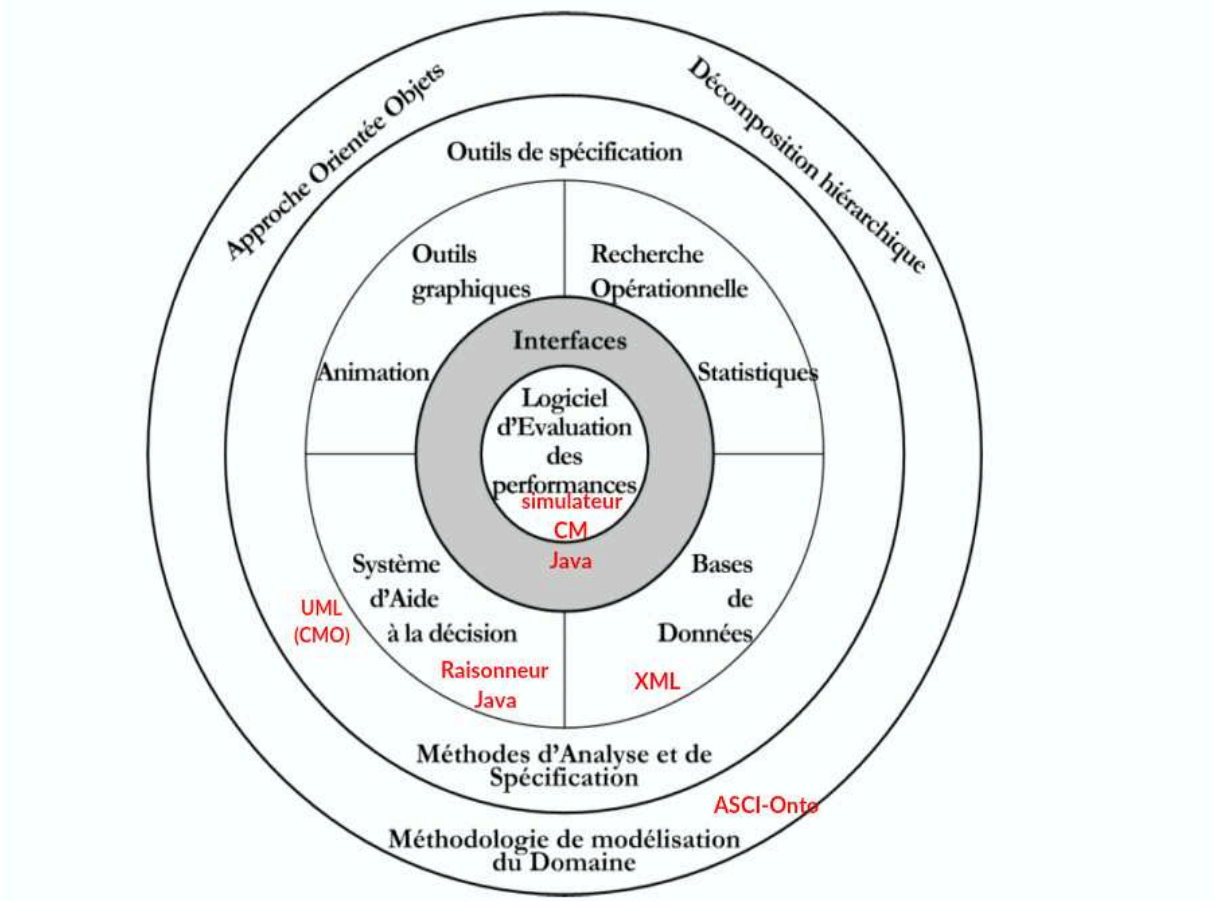


FIGURE 6.11 – *Environnement logiciel ASCI-Onto*

6.4.3 Implémentation des classes de la CMO

Comme expliqué dans la section précédent, le fichier XML est une base de données regroupant les informations concernant les fournisseurs, services, ressources, etc. par exemple, la figure 6.12 représente la description du service3 dans le langage OWL. Afin d'obtenir ce fichier, nous avons raisonné sur le formalisme proposé en OWL afin d'extraire le graphe RDF inféré.

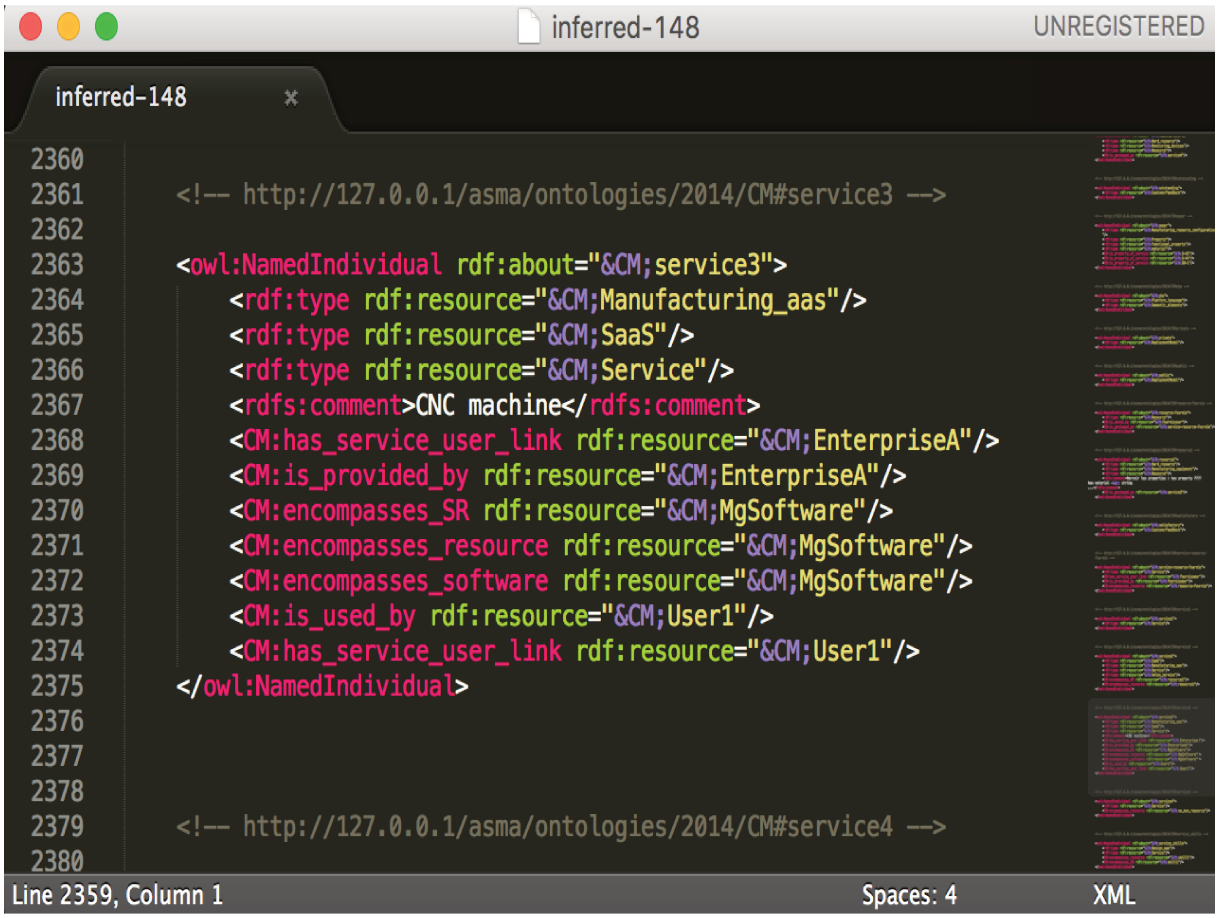


FIGURE 6.12 – *Description OWL du service 3*

Nous l'avons ensuite transformé en une base de données XML en nous basant sur les description données dans le fichier inféré. La figure 6.13 est une représentation des services disponibles dans la base. Un service est décrit entre deux balises : <Individual> et </Individual>. La description du service comporte : son type (la classe à laquelle il appartient), la ressource qui permet de l'exécuter, l'identifiant du fournisseur et l'URI de cette dernière.

services.xml

```xml
<Individuals>
    <Individual uid="service2">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>resource1</Resource>
        <Provider>EnterpriseA</Provider>
        <uri>http://localhost:8888/CNC/CNC_Simulator.php</uri>
    </Individual>

    <Individual uid="service3">
        <Type>SaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>MgSoftware</Resource>
        <Provider>EnterpriseB</Provider>
        <uri>uri2</uri>
    </Individual>

    <Individual uid="S-A1">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>MgSoftware</Resource>
        <Provider>EnterpriseB</Provider>
        <uri>uri3</uri>
    </Individual>

    <Individual uid="S-A2">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>machineTool-A</Resource>
        <Provider>company-X</Provider>
        <uri>uri4</uri>
    </Individual>

    <Individual uid="S-A2">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>machineTool-A</Resource>
        <Provider>company-X</Provider>
        <uri>uri4</uri>
    </Individual>

</Individuals>
```

FIGURE 6.13 – *Exemple de services décrits dans la base de données XML*

En nous basant sur l'architecture définie précédemment, les classes *Consumer* et *Provider* représentant les acteurs du système sont implémentées de la manière suivante :

```
public class Consumer {

private String firstName;
private String lastName;
private String address;
private String phone;
private String enterprise;
private String status;
protected int Id;

```

```
11 public Consumer(String firstName,String lastName, String
      address, String phone){
12 this.firstName = firstName;
13 this.lastName = lastName;
14 this.address = address;
15 this.phone = phone;
16 this.status = "person";
17         }
18
19 public Consumer(String firstName,String lastName, String
      address, String phone, String enterprise){
20 this.firstName = firstName;
21 this.lastName = lastName;
22 this.address = address;
23 this.phone = phone;
24 this.enterprise = enterprise;
25 this.status = "enterprise";
26
27         }
28
29 public int getId(){
30 return Id;
31         }
32
33 public void setId(int id){
34 this.Id = id;
35         }
36         }
```

Classe Provider :

```
1  public class Provider {
2
3  private String firstName;
4  private String lastName;
5  private String address;
6  private String phone;
7  private String enterprise;
8  private String status;
9  protected int Id;
10 public Provider(String firstName,String lastName, String
      address, String phone){
11 this.firstName = firstName;
12 this.lastName = lastName;
13 this.address = address;
14 this.phone = phone;
15 this.status = "person";
16 }
17
```

```
18 public Provider(String firstName,String lastName, String
      address, String phone, String enterprise){
19 this.firstName = firstName;
20 this.lastName = lastName;
21 this.address = address;
22 this.phone = phone;
23 this.enterprise = enterprise;
24 this.status = "enterprise";
25
26 }
27
28 public int getId(){
29 return Id;
30
31 }
32
33 public void setId(int id){
34 this.Id = id;
35 }
36
37 }
```

Nous avons défini la classe *Engine* qui servira de parser de la base de données XML comme suit :

```
1  package org.cloudbus.cloudsim;
2
3  import java.io.File;
4  import java.io.IOException;
5  import java.util.ArrayList;
6
7  import javax.xml.parsers.DocumentBuilder;
8  import javax.xml.parsers.DocumentBuilderFactory;
9  import javax.xml.parsers.ParserConfigurationException;
10
11 import org.w3c.dom.Document;
12 import org.w3c.dom.Element;
13 import org.w3c.dom.Node;
14 import org.w3c.dom.NodeList;
15 import org.xml.sax.SAXException;
16
17 public class Engine {
18
19 /*
20 * Déclaration des vecteurs utilisé durant le traitement
21 */
22
23 public static final ArrayList<String[]> services = new
      ArrayList();
```

```
24
25 public static void Parser() {
26 /*
27 * Etape 1 : Récupération d'une instance de la classe
      "DocumentBuilderFactory"
28 */
29 final DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
30
31 try {
32 /*
33 * Etape 2 : Création d'un parseur
34 */
35 final DocumentBuilder builder = factory.newDocumentBuilder();
36
37 /*
38 * Etape 3 : Création d'un Document
39 */
40
41 Document document= builder.parse(new File("services.xml"));
42
43 //Affichage du prologue
44 System.out.println("*************PROLOGUE************");
45 System.out.println("version : " + document.getXmlVersion());
46 System.out.println("encodage : " + document.getXmlEncoding());
47 System.out.println("standalone : " +
      document.getXmlStandalone());
48
49 /*
50 * Etape 4 : Récupération de l'Element racine
51 */
52 final Element racine = document.getDocumentElement();
53
54
55 //Affichage de l'élément racine
56 System.out.println("\n*************RACINE************");
57 System.out.println(racine.getNodeName());
58 System.out.println(racine.getFirstChild().getNodeName());
59
60 /*
61 * Etape 5 : Récupération des noeuds
62 */
63
64 final NodeList racineNoeuds = racine.getChildNodes();
65 final int nbRacineNoeuds = racineNoeuds.getLength();
66
67 for (int p = 0; p<nbRacineNoeuds; p++) {
68 // Récupération de tous les individus de type services
```

```
69 if(racineNoeuds.item(p).getNodeName().contentEquals("Individual")){
70 String tmpoc[] = new String[7];
71 ArrayList<String> tmp = new ArrayList();
72 final Element individu = (Element) racineNoeuds.item(p);
73
74 tmpoc[0] =  individu.getAttribute("uid");
75
76 NodeList nodeList = racineNoeuds.item(p).getChildNodes();
77 for (int i = 0; i < nodeList.getLength(); i++) {
78 Node currentNode = nodeList.item(i);
79 if ((currentNode.getNodeType() ==
      Node.ELEMENT_NODE)&&(currentNode.getNodeName().contentEquals("Type"))
      ) {
80 tmp.add(currentNode.getTextContent());
81
82 }
83 }
84 for(int f=0; f<tmp.size();f++){
85 tmpoc[f+1] = tmp.get(f);
86 }
87 tmpoc[4] =
      individu.getElementsByTagName("Resource").item(0).getTextContent();
88 tmpoc[5] =
      individu.getElementsByTagName("Provider").item(0).getTextContent();
89 tmpoc[6] =
      individu.getElementsByTagName("uri").item(0).getTextContent();
90
91 services.add(tmpoc) ;
92 }
93 }
94 System.out.println("Liste de services disponibles :");
95 for(int i = 0; i<services.size();i++){
96 System.out.println( services.get(i)[0] + " de type :  " +
      services.get(i)[1] + " , " + services.get(i)[2] + " et " +
      services.get(i)[3] + "  Fourni par  "+ services.get(i)[5]);
97 }
98
99 }catch (final ParserConfigurationException e) {
100 e.printStackTrace();
101 }
102 catch (final SAXException e) {
103 e.printStackTrace();
104 }
105 catch (final IOException e) {
106 e.printStackTrace();
107 }
108 }
109 }
```

Enfin la classe principale *CM* qui permet d'instancier les classes ci-dessous :

```
1  package org.cloudbus.cloudsim;
2
3  import java.awt.Desktop;
4  import java.io.IOException;
5  import java.net.URI;
6  import java.util.Scanner;
7
8  public class CM {
9
10 public static void main(String[] args) {
11 // TODO Auto-generated method stub
12 String uri_service = "";
13 // Appel de la méthode Parser implémentée dans la classe Engine
14 Engine.Parser();
15 //Afficher à l'utilisateur la liste des services disponibles
16 Scanner sc = new Scanner(System.in);
17 String str2 = sc.nextLine();
18 System.out.println("\nVous avez choisi le service : " + str2 +
      "\n");
19 //Récupérer l'URI du service choisi par l'utilisateur
20 for(int y=0; y<Engine.services.size();y++){
21 if(Engine.services.get(y)[0].equalsIgnoreCase(str2)){
22 uri_service = Engine.services.get(y)[6];
23 }
24 }
25 try {
26 //Lancement de la connexion et affichage de la page
      représentant le service
27 URI uri = URI.create(uri_service);
28
29 Desktop.getDesktop().browse(uri);
30 }
31 catch (IOException ex){
32 ex.printStackTrace();
33 System.out.println("Ouverture échouée");
34 }
35 }
36 }
```

6.5 Illustration du fonctionnement d'une plate-forme CM en utilisant la méthodologie proposée : ASCI-Onto

6.5.1 Présentation de l'exemple

Notre exemple consiste à utiliser les classes implémentées et décrites dans la section précédente pour représenter une situation où l'utilisateur s'adresse à la plate-forme afin de choisir le service dont il a besoin parmi un ensemble de propositions. Les services proposés sont décrits dans un fichier XML mais sont hébergés chez les fournisseurs qui proposent un accès distant à ces derniers. Pour notre simulation, nous avons hébergé les services dans des serveurs locaux type WAMP.

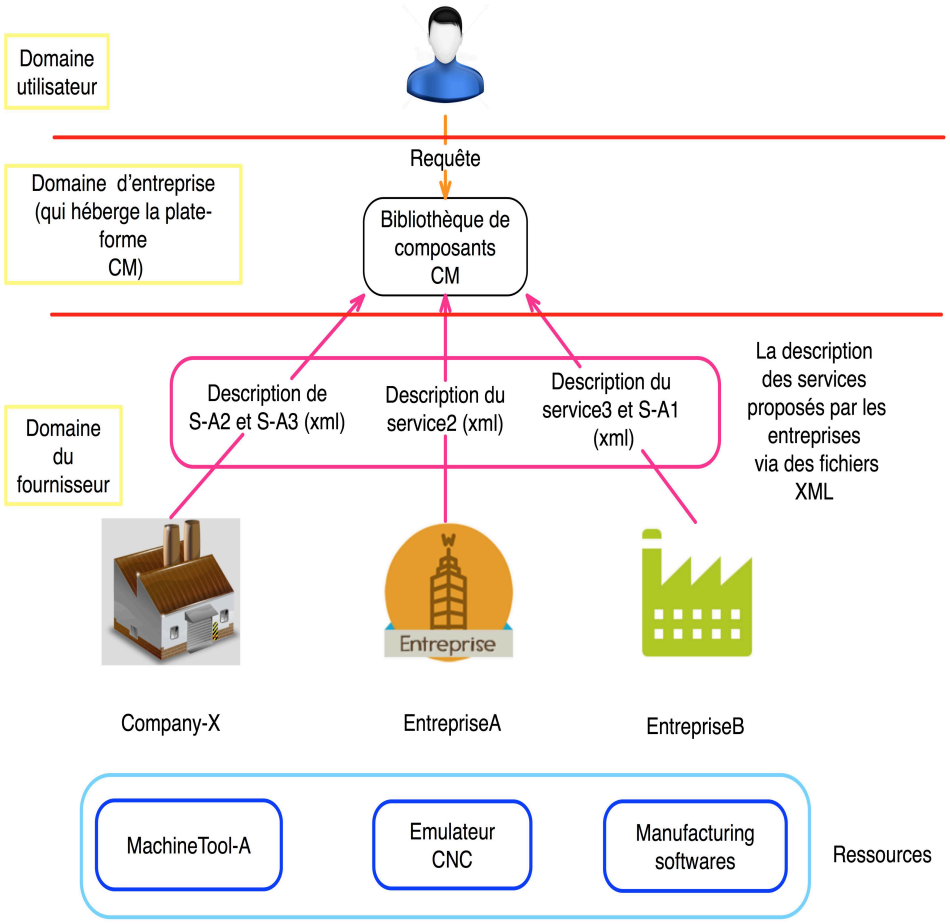


Figure 6.14 – *Présentation de l'exemple étudié et sa structure par rapport à l'achitecture de [Xu 12b]*

6.5.2 Le modèle de connaissance

Le modèle de connaissance représente les classes que nous avons utilisées pour notre exemple. Le modèle de connaissance est dérivé à partir du modèle générique de connaissance (construit durant les phases *Analyse-Validation* d'ASCI-Onto) et consiste

en le recueil des informations liées au système étudié. Les classes utilisées pour notre scénario sont présentées dans la figure 6.15.

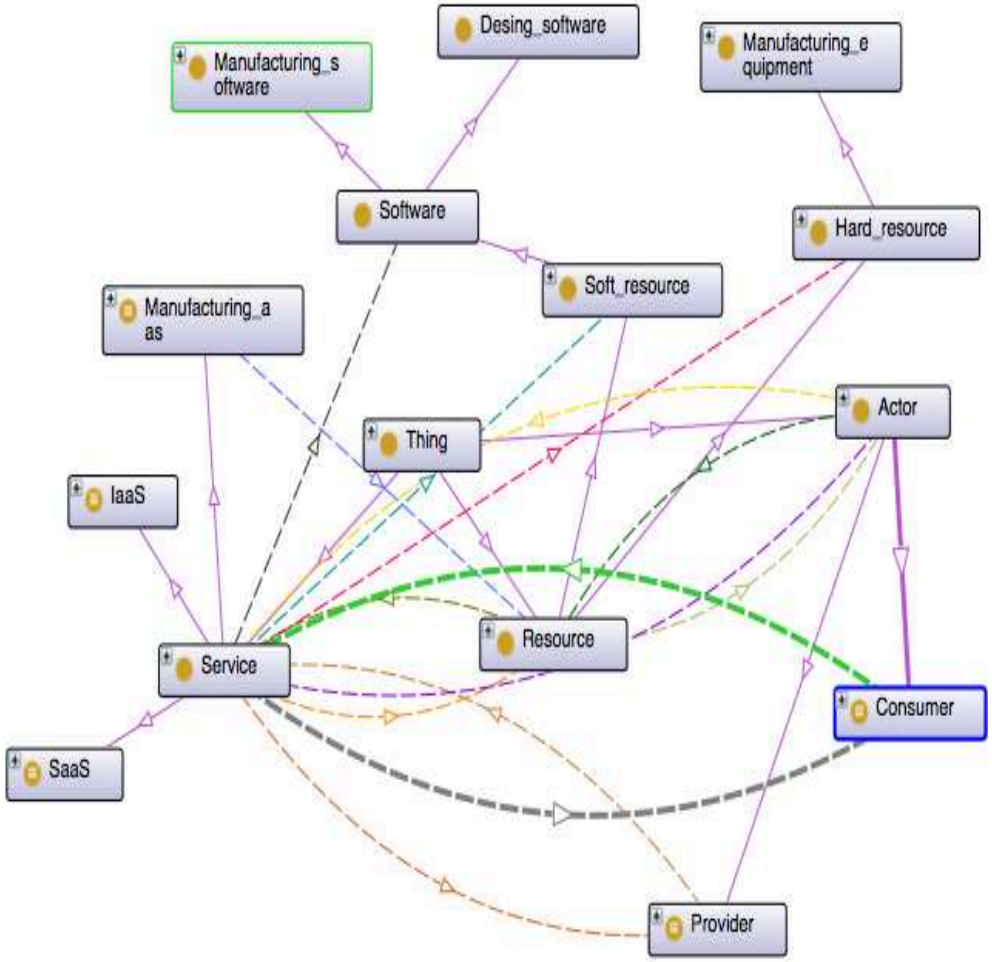


FIGURE 6.15 – *Elaboration du modèle de connaissance du système étudié à partir du Modèle générique de connaissance*

6.5.3 Le modèle d'action

Afin de simuler la machine outil, nous avons utilisé l'émulateur CNC développé par William HILTON[4] qui a developpé un émulateur d'une machine de type DYNA MYTE 2400 quand il était doctorant à l'université de Drexel à Philadelphie. Nous avons installé un serveur MAMP[5] afin d'hébérger le service représentant la machine à commandes numériques. La figure ci-dessous indique que le serveur est démarré (via l'interface graphique ou la ligne de commande).

4. https://sites.google.com/site/wmhilton/home consulté en Janvier 2016
5. https://www.mamp.info/en/ consulté en Janvier 2016

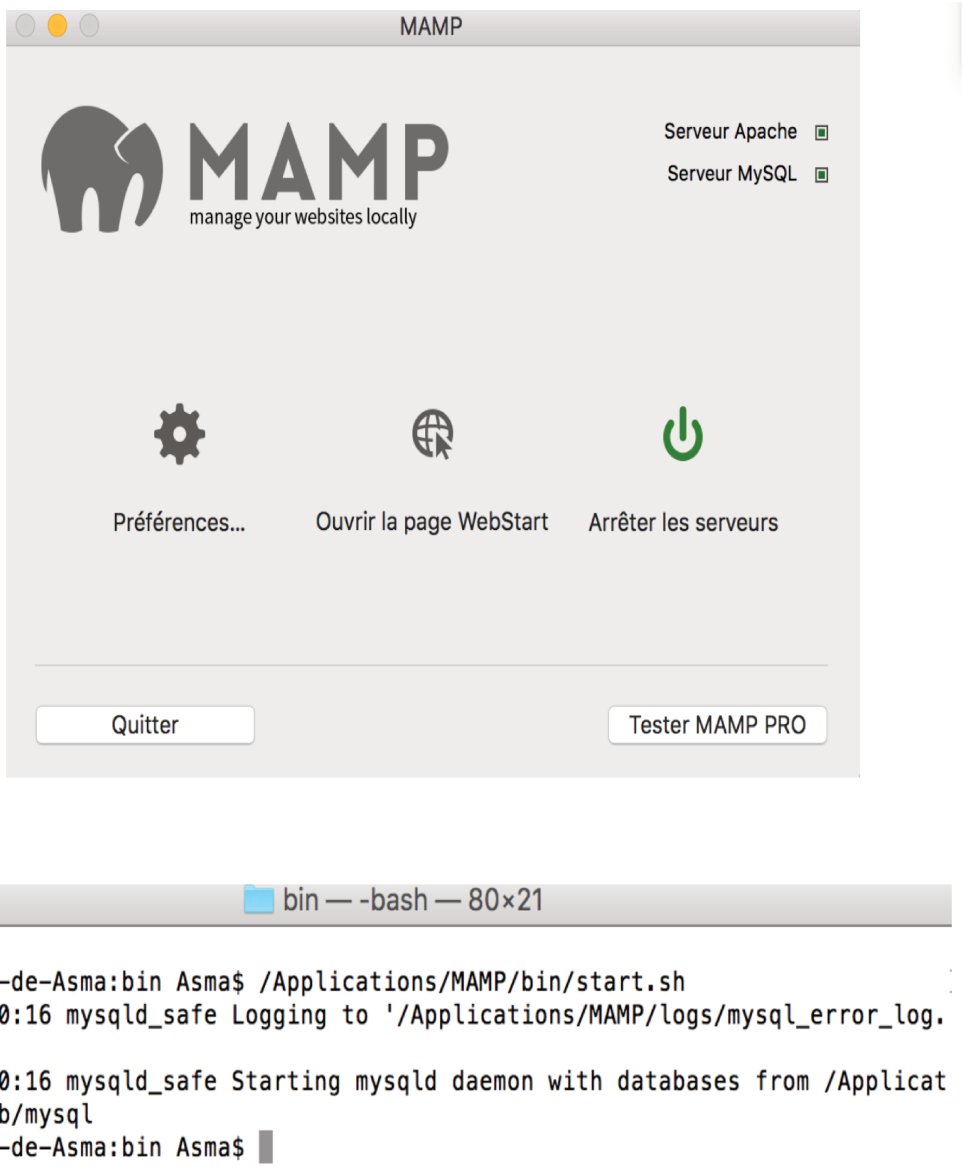


FIGURE 6.16 – *Serveur MAMP démarré*

Le résultat de l'exécution de la classe principale *CM* est le suivant :

```
1  **************PROLOGUE************
2  version : 1.0
3  encodage : null
4  standalone : false
5
6  **************RACINE************
7  Individuals
8
9  Liste de services disponibles :
10 service2 de type :  IaaS , Manufacturing_aas et Service  Fourni
       par  EnterpriseA
```

```
service3 de type :  SaaS , Manufacturing_aas et Service  Fourni
   par  EnterpriseB
S-A1 de type :  IaaS , Manufacturing_aas et Service  Fourni par
    EnterpriseB
S-A2 de type :  IaaS , Manufacturing_aas et Service  Fourni par
    company-X
S-A2 de type :  IaaS , Manufacturing_aas et Service  Fourni par
    company-X
service2

Vous avez choisi le service : service2
```

La classe *CM* a parsé le fichier XML contenant la description des services et a affiché pour chaque service son nom, son type, et le nom du fournisseur. L'étape suivante est le choix de l'utilisateur, qui saisit le nom du service auquel il veut accéder. Dans notre exemple, l'utilisateur a choisi *service2*.
La page web représentant le service2 choisi dans l'exemple a été développée en utilisant HTML5 et JavaScript, son url est la suivante : `http://localhost:8888/CNC/CNC_Simulator.php`. La figure 6.17 montre la page d'accueil qui donne accès au service et à l'utilisation et l'exploitation de la machine CNC via son interface de commande.

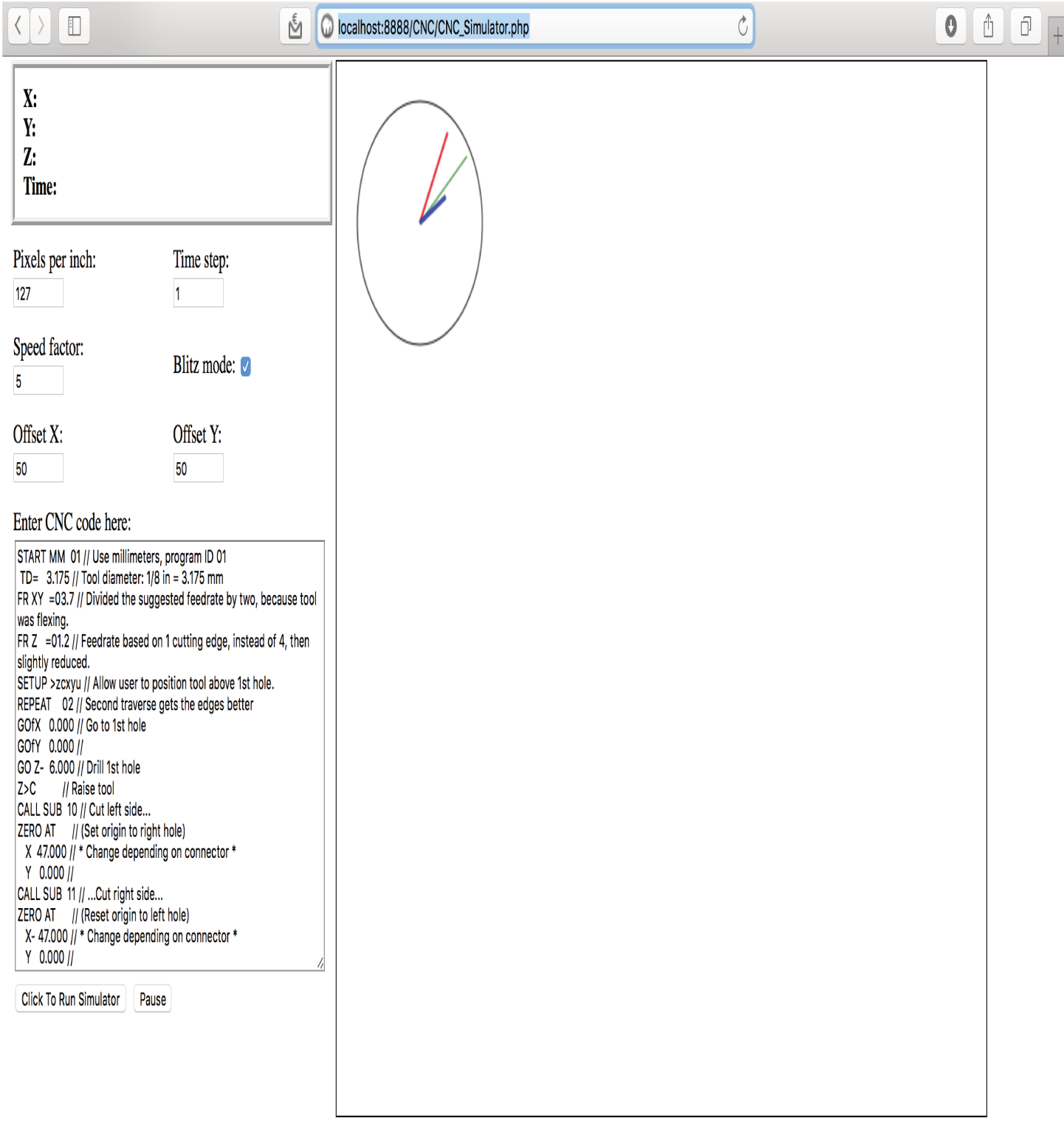


FIGURE 6.17 – *Page d'accueil du service 2 - L'accès se fait via un navigateur Web (Safari)*

Nous avons utilisé le code ci-dessous (figure 6.18), fourni par le développeur de l'émulateur et qui représente un ensemble d'indication pour le déplacement de la fraise selon des coordonnées X,Y et Z. Chaque ligne représente la position de la fraise. Par exemple, les lignes : 7, 8, 9 indiquent que, pour effectuer le premier point sur la pièce, la fraise doit se déplacer d'une valeur de 8.5 sur l'axe des abscisses, 121 sur l'axe des ordonnées, et doit être insérer dans la pièce avec une profondeur équivalant à 9.

```
START MM  02 // Use millimeters. Program ID 01 is taken by the D-SUB program I think.
 TD=   3.175 // Tool diameter: 1/8 in = 3.175 mm
FR XY  =03.7 // Divided the suggested feedrate by two, because tool was flexing.
FR Z   =01.2 // Feedrate based on 1 cutting edge, instead of 4, then slightly reduced.
SETUP >zcxyu // Allow user to position tool above bottom-left corner.
Z>C          // Raise tool
GOfX   8.500 // Point 1 of Figure 1
GOfY 121.000 //
GO Z-  9.000 // Insert tool
GO X   8.000 // Point 2
   Y 119.000 //
GO X   9.000 // Point 3
   Y 110.500 //
GO X   6.500 // Point 4
   Y 100.500 //
GO X  12.500 // Point 5
   Y  92.500 //
GO X  13.500 // Point 6
   Y  83.500 //
GO X  25.000 // Point 7
   Y  70.500 //
GO X  25.500 // Point 8
   Y  64.300 //
GO X  28.000 // Point 9
   Y  60.500 //
GO X  30.000 // Point 10
   Y  60.500 //
GO X  32.000 // Point 11
   Y  57.500 //
GO X  30.000 // Point 12
   Y  55.000 //
GO X  31.000 // Point 13
   Y  51.500 //
GO X  43.500 // Point 14
   Y  31.000 //
GO X  53.000 // Point 15
   Y  30.000 //
GO X  63.000 // Point 16
   Y  25.500 //
GO X  74.000 // Point 17
   Y  16.000 //
GO X  76.500 // Point 18
   Y   9.000 //
GO X 100.000 // Point 19
   Y  11.000 //
GO X  99.500 // Point 20
   Y  21.000 //
GO X 103.500 // Point 21
   Y  28.000 //
GO X  99.000 // Point 22
   Y  36.000 //
GO X  47.500 // Point 23
   Y  83.500 //
GO X  47.500 // Point 24
   Y 121.000 //
GO X   8.500 // Return to Point 1
   Y 121.000 //
Z>C          // Raise tool

GOfX   0.000 // Return home
GOfY   0.000 //
GO Z-  9.000 // Insert tool
GO X 144.000 // Point 2 of border
GO Y 126.000 // Point 3 of border
GO X   0.000 // Point 4 of border
GO Y   0.000 // Return home.
Z>C          // Raise tool
END NEW PART //
```

FIGURE 6.18 – *Code permettant de manipuler l'émulateur*

Une fois ce code exécuté, l'utilisateur peut suivre les étapes de son exécution et donc de la découpe, et est informé en temps réel du temps de l'exécution et du temps final comme le montrent les figures qui suivent.

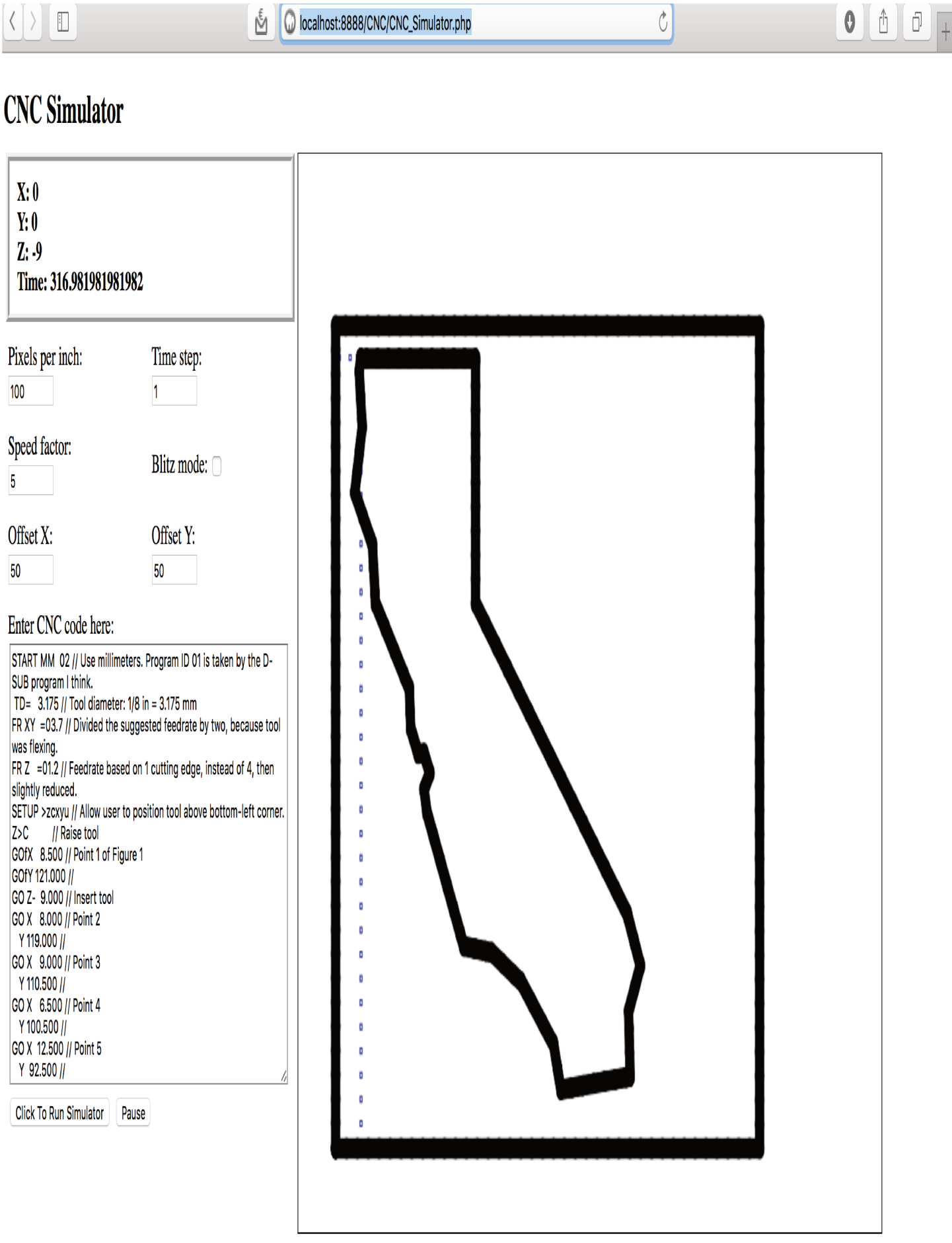
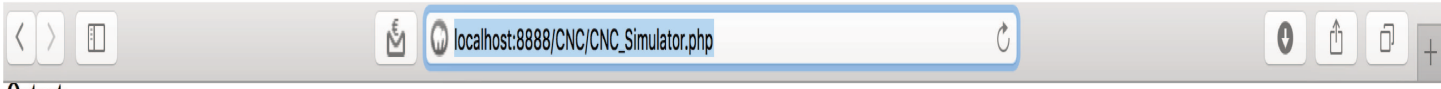


FIGURE 6.19 – *Résultats de la simulation*

6.5.4 Modèle de résultat

Comme défini précédemment (section 4.3.3.3), le modèle de résultats est alimenté à partir du modèle d'action. Il doit permettre de répondre aux objectifs fixés au début du processus d'instanciation sur un système du domaine. Il regroupe les résultats quantitatifs et qualitatifs fournis par le modèle d'action mis en forme pour permettre l'analyse du système et la prise de décision (indicateurs de performances, tableaux de bords, représentations graphiques). Ce modèle doit être conçu dans un souci d'évolutivité car en fonction des résultats obtenus, les décideurs vont pouvoir évaluer la pertinence des solutions envisagées et choisir ou non de modifier la structure de leur système (décomposition systémique), leurs processus (comportements des entités) ou leurs règles de gestion. Cette modification est illustrée dans la figure 6.1 par la boucle, liant « Modèle de résultats » et « système du domaine » car elle représente les actions à appliquer sur ce dernier. Une fois ces actions appliquées, le modèle de connaissance est par conséquent modifié, et un nouveau modèle d'action pourra être généré afin d'obtenir de nouveaux résultats permettant au décideur de constater les incidences des modifications apportées sur les critères de performance.

Par exemple, dans la figure 6.20, chaque ligne représente le temps pris par la machine outil pour effectuer les différents déplacements de la fraise. À partir de ces résultats, l'utilisateur peut effectuer des modifications mineures, c'est-à-dire avoir le choix entre : (i) conserver le paramétrage de la machine et donc continuer à utiliser ce service, (ii) modifier simplement le paramétrage de la machine outil afin d'obtenir un meilleurs temps d'exécution ou bien (iii) choisir un autre service. Un exemple de modification majeure peut être les changements appliqués au système étudié au début de ce processus pour générer un autre modèle de connaissance, un autre modèle d'action afin d'obtenir d'autres résultats, qui cette fois-ci ne dépendent pas du service choisi, mais des décisions prises à partir du système de départ.



Output:

```
There are 68 lines

Line 0: Unit: MM... Millimeters... Scale: 3.937007874015748
Line 1: Tool Diameter:    3.175... ctx.lineWidth = 3.174999952316284
Line 2: Setting feed rate for XY  to 03.7...  set FR X = 3.7 set FR Y = 3.7
Line 3: Setting feed rate for Z   to 01.2...  set FR Z = 1.2
Line 4: Setup: zcxyu Note: in real life, this command would let you position the tool at a new 'zero' position.
Line 5: Z>C: Z = 2
Line 6: Go... X coor: 8.5... Time Required: 2.2972972972972974... X inc: 3.6999999999999997 Y inc: 0 Z inc: 0...
Line 7: Go... Y coor: 121... Time Required: 32.7027027027027... X inc: 0 Y inc: 3.7 Z inc: 0.............................
Line 8: Go... Z coor: -9... Time Required: 9.166666666666668... X inc: 0 Y inc: 0 Z inc: -1.2.........
Line 9: Go... X coor: 8... Y coor: 119... Time Required: 0.5405405405405405... X inc: -0.9250000000000002 Y inc: -3.7000000000000006 Z inc: 0.
Line 11: Go... X coor: 9... Y coor: 110.5... Time Required: 2.2972972972972974... X inc: 0.43529411764705883 Y inc: -3.6999999999999997 Z inc: 0...
Line 13: Go... X coor: 6.5... Y coor: 100.5... Time Required: 2.7027027027027026... X inc: -0.925 Y inc: -3.7 Z inc: 0...
Line 15: Go... X coor: 12.5... Y coor: 92.5... Time Required: 2.162162162162162... X inc: 2.7750000000000004 Y inc: -3.7000000000000006 Z inc: 0...
Line 17: Go... X coor: 13.5... Y coor: 83.5... Time Required: 2.4324324324324325... X inc: 0.4111111111111111 Y inc: -3.7 Z inc: 0...
Line 19: Go... X coor: 25... Y coor: 70.5... Time Required: 3.513513513513513... X inc: 3.2730769230769234 Y inc: -3.7 Z inc: 0....
Line 21: Go... X coor: 25.5... Y coor: 64.3... Time Required: 1.6756756756756763... X inc: 0.29838709677419345 Y inc: -3.7 Z inc: 0..
Line 23: Go... X coor: 28... Y coor: 60.5... Time Required: 1.0270270270270263... X inc: 2.434210526315791 Y inc: -3.6999999999999997 Z inc: 0..
Line 25: Go... X coor: 30... Y coor: 60.5... Time Required: 0.5405405405405405... X inc: 3.7000000000000006 Y inc: 0 Z inc: 0.
Line 27: Go... X coor: 32... Y coor: 57.5... Time Required: 0.8108108108108107... X inc: 2.466666666666667 Y inc: -3.7 Z inc: 0.
Line 29: Go... X coor: 30... Y coor: 55... Time Required: 0.6756756756756757... X inc: -2.96 Y inc: -3.7 Z inc: 0.
Line 31: Go... X coor: 31... Y coor: 51.5... Time Required: 0.9459459459459459... X inc: 1.0571428571428572 Y inc: -3.7 Z inc: 0.
Line 33: Go... X coor: 43.5... Y coor: 31... Time Required: 5.54054054054054... X inc: 2.25609756097561 Y inc: -3.7 Z inc: 0......
Line 35: Go... X coor: 53... Y coor: 30... Time Required: 2.5675675675675675... X inc: 3.7 Y inc: -0.3894736842105263 Z inc: 0...
Line 37: Go... X coor: 63... Y coor: 25.5... Time Required: 2.7027027027027026... X inc: 3.7 Y inc: -1.665 Z inc: 0...
Line 39: Go... X coor: 74... Y coor: 16... Time Required: 2.972972972972973... X inc: 3.7 Y inc: -3.1954545454545458 Z inc: 0...
Line 41: Go... X coor: 76.5... Y coor: 9... Time Required: 1.8918918918918919... X inc: 1.3214285714285714 Y inc: -3.7 Z inc: 0..
Line 43: Go... X coor: 100... Y coor: 11... Time Required: 6.351351351351351... X inc: 3.7 Y inc: 0.3148936170212766 Z inc: 0.......
Line 45: Go... X coor: 99.5... Y coor: 21... Time Required: 2.7027027027027026... X inc: -0.185 Y inc: 3.7 Z inc: 0...
Line 47: Go... X coor: 103.5... Y coor: 28... Time Required: 1.8918918918918919... X inc: 2.1142857142857143 Y inc: 3.7 Z inc: 0..
Line 49: Go... X coor: 99... Y coor: 36... Time Required: 2.162162162162162... X inc: -2.0812500000000003 Y inc: 3.7000000000000006 Z inc: 0...
Line 51: Go... X coor: 47.5... Y coor: 83.5... Time Required: 13.918918918918918... X inc: -3.7 Y inc: 3.4126213592233015 Z inc: 0.............
Line 53: Go... X coor: 47.5... Y coor: 121... Time Required: 10.135135135135135... X inc: 0 Y inc: 3.7 Z inc: 0...........
Line 55: Go... X coor: 8.5... Y coor: 121... Time Required: 10.54054054054054... X inc: -3.7 Y inc: 0 Z inc: 0...........
Line 57: Z>C: Z = 2
Line 58: Skipping line:
Line 59: Go... X coor: 0... Time Required: 2.2972972972972974... X inc: -3.6999999999999997 Y inc: 0 Z inc: 0...
Line 60: Go... Y coor: 0... Time Required: 32.7027027027027... X inc: 0 Y inc: -3.7 Z inc: 0.............................
Line 61: Go... Z coor: -9... Time Required: 9.166666666666668... X inc: 0 Y inc: 0 Z inc: -1.2.........
Line 62: Go... X coor: 144... Time Required: 38.91891891891892... X inc: 3.7 Y inc: 0 Z inc: 0.................................
Line 63: Go... Y coor: 126... Time Required: 34.05405405405405... X inc: 0 Y inc: 3.7000000000000006 Z inc: 0.............................
Line 64: Go... X coor: 0... Time Required: 38.91891891891892... X inc: -3.7 Y inc: 0 Z inc: 0.................................
Line 65: Go... Y coor: 0... Time Required: 34.05405405405405... X inc: 0 Y inc: -3.7000000000000006 Z inc: 0.............................
Line 66: Z>C: Z = 2
Line 67: Encountered an END
Ended because there are no more lines.
canvas_stack = 2
ctx.restore()
```

FIGURE 6.20 – *Compte rendu de la simulation*

6.6 Conclusion

Ce chapitre a présenté les deux étapes qui ont permis la conception et l'implémentation d'une bibliothèque de composants réutilisables d'une plate-forme CM. La première phase consiste à expliquer le lien entre le modèle sémantique, présenté dans le chapitre précédent (CMO) et l'architecture finale de la plate-forme. Nous avons démontré qu'il y a besoin de dériver notre ontologie CMO en un diagramme UML afin de représenter l'état statique à l'aide des classes du système finale. Cette dérivation est basée sur des travaux de la littérature expliquant les équivalences entre le langage ontologique OWL et UML. Durant cette même étape de conception, nous avons expliqué comment modifier l'environnement de simulation de CC, en le fusionnant avec le diagramme de classes UML pour proposer une environnement de simulation CM. Ensuite, l'architecture envisagée pour cet objectif présentée ainsi que certains modules de la bibliothèque de composants. Nous avons expliqué le fonctionnement de notre proposition par un cas d'étude montrant comment un utilisateur peut accéder à un service hébergé sur un serveur local, l'exploiter et les résultats obtenus. L'implémentation de la plate-forme CM n'est pas finie, et cette dernière est toujours en cours de développent. Nous présentons dans ce qui suit notre contribution, les limites de notre implémentation actuelle ainsi que les perspectives de recherche.

Conclusion et perspectives de recherche

6.7 Conclusion générale

L'objectif général de nos travaux de thèse était de définir une méthodologie qui permette de mettre en place une plate-forme *Cloud Manufacturing* servant de support aux systèmes industriels traditionnels pour migrer vers le Cloud. L'origine de ces travaux réside dans le constat suivant : les études des systèmes dans un contexte PLM font apparaître des problèmes nouveaux, plus particulièrement, les systèmes actuels manquent de structures et de procédures adaptatives qui seraient capables de supporter une utilisation et une allocation dynamiques, flexibles et à la demande des ressources afin de suivre les exigences du marché. Ces problèmes ne peuvent pas être résolus simplement par l'augmentation du nombre de ressources ni par l'extension des heures de travail pour suivre la demande. Nous étions confrontés au début de notre travail à une difficulté découlant du nombre réduit d'articles traitant du domaine CM. Étant un nouveau paradigme, le CM apparaissait principalement dans des études menées dans des universités chinoises et néo-zélandaises. On remarque par ailleurs que ces travaux introduisent des définitions du CM et discutent la vision stratégique qu'il offre.

Pour atteindre cet objectif général, nous avons répondu à un ensemble de sous-problématiques et pensons apporter les contributions suivantes :

— nous avons proposé un point de vue nouveau sur les SIE en introduisant une vision orientée « service » ;
— nous avons proposé un modèle générique basé sur les ontologies (la CMO) permettant d'intégrer les différents fournisseurs de services, les utilisateurs, les services, les ressources et tout autre concept lié au CM. L'expressivité des ontologies, fondées sur des concepts porteurs de sens, permet le partage des informations qui est primordial dans des environnements distribués comme le CM. De plus, l'inférence permet de proposer un modèle intelligent capable de déduire des informations implicites ;
— nous avons proposé une conception d'une plate-forme CM basée sur le modèle unifié CMO en utilisant un simulateur de CC. Durant cette phase, nous avons proposé une adaptation du simulateur au domaine CM ;
— nous avons proposé une méthodologie de modélisation ASCI-Onto pour le domaine CM dans laquelle nous avons détaillé l'enchainement nécessaire d'étapes, depuis la formalisation du modèle générique de connaissance et sa validation jusqu'à la conception d'une plate-forme CM. Cette méthodologie peut être réutilisée pour modéliser le même domaine (CM) mais en utilisant d'autres outils et langages

de programmation pour produire le modèle générique de connaissance et la bibliothèque de composants réutilisables. Elle peut aussi être réutilisée avec les mêmes outils (Ontologies, Java) pour modéliser un autre domaine ainsi que les systèmes qui le composent.

~

Le travail relatif à l'état de l'art a fait l'objet d'une publication scientifique [Talhi 13]. Le modèle générique de connaissance CMO a été validé scientifiquement [Talhi 14], de même que le cadre méthodologique ASCI-Onto [Talhi 15c]. Enfin, une publication de notre réflexion sur l'étape de conception de la bibliothèque de composants a été réalisée [Talhi 15a, Talhi 15b].

6.8 Validation des questions de recherche

Nous pouvons, à la vue de nos résultats, valider notre première question de recherche :

Comment modéliser le domaine du Cloud Manufacturing ?

Nous avons proposé un modèle générique de connaissance CMO pour représenter le domaine CM basé sur les ontologies. Nous avons montré que l'apport des ontologies à ce niveau consiste en la richesse sémantique et la capacité d'inférence qu'elles proposent. Ces deux avantages nous ont permis de développer un modèle porteur de sens et intelligent. D'autre part, la capacité d'inférence permet la découverte de nouvelles informations à partir de celles définies dans le modèles et ainsi réduire l'engagement de modélisation, ce qui facilite la maintenance du modèle et son intégration. Les concepts qui composent l'ontologie ont été définis à partir d'un état de l'art sur les modèles CC et les modèles manufacturiers. Nous avons validé notre proposition via un scénario issu de la littérature scientifique.

Concernant la deuxième question, nous estimons que nos travaux la valident en partie.

Comment implémenter une plate-forme Cloud Manufacturing ?

Nous avons défini une méthodologie, ASCI-Onto, qui englobe la première question de recherche, première étape vers la mise en place d'une plate-forme CM. Durant la deuxième étape de la méthodologie, nous avons proposé une conception et une implémentation d'une bibliothèque de composants réutilisables à partir de la CMO. Nous avons décrit l'architecture envisagée et avons présenté un exemple de fonctionnement d'une plate-forme CM. Cette validation est partielle car elle ne porte pas sur un cas réel industriel avec une grande quantité de données et de ressources proposées. D'autre part, nous n'avons appliqué notre méthodologie à un seul système du domaine étudié. La validation de cette deuxième question nécessite l'amélioration et le développement complet de l'architecture proposée et l'application de la méthodologie sur plusieurs systèmes afin de montrer comment ASCI-Onto peut être un réel support d'aide à la décision et un guide pour assister les entreprises dans leur processus de migration vers le cloud.

6.9 Les perspectives de recherche

Il est important de préciser que notre travail propose un ensemble de réflexions constituant la première étape vers la mise en place d'une plate-forme CM robuste. Les principales perspectives de recherche qui apparaissent à l'issue de cette thèse portent sur les questions de recherche restées ouvertes et qu'il conviendrait d'explorer afin de compléter ce travail.
— Utiliser la méthodologie avec les ontologies sur un autre domaine.
— Appliquer sur un cas industriel complet.

6.9.1 Amélioration de la proposition et limite de l'implémentation

Nous avons effectué un état de l'art afin de recenser les concepts liés au CM. Néanmoins l'ontologie proposée demeure non exhaustive et certains concepts peuvent être ajoutés selon la façon dont on souhaite l'utiliser. Dans notre cas, l'ontologie est utilisée comme intermédiaire entre les fournisseurs et les clients afin de servir de support à ces derniers dans le choix des services. A ce stade, l'utilisateur connait par avance le workflow a exécuter et cherche via la plate-forme les services lui permettant d'accomplir ses tâches. Par conséquent, notre ontologie ne comporte pas le terme **Processus**. Comme nous l'avons expliqué précédemment, les outils d'implémentation d'ontologies ne sont pas matures pour permettre une implémentation complète intégrant la partie raisonnement. Les tests effectués sur l'ontologie ont permis d'avoir la liste des services répondant à une requête de l'utilisateur grâce au raisonneur mais ce dernier n'a pas la capacité de choisir parmi ces services lequel est le plus adéquat pour l'utilisateur. D'autre part, le nombre de réponses retournées dépend du nombre d'entités présentes dans le système. Sachant qu'un environnement CM peut comporter des millions d'instances, il devient impossible pour l'utilisateur de choisir parmi des milliers de services. Cette limite du modèle nous amène à considérer les méthodes issues de la recherche opérationnelle pour l'implémentation du raisonneur. En effet, ces méthodes et algorithmes permettent d'optimiser les réponses et les adapter en fonction de l'objectif de l'utilisateur (réduction des coûts, de temps de traitement, etc) et les contraintes du systèmes. Il existe de nombreux travaux sur la composition de services qui ont suivi l'émergence de l'architecture orientée service (SOA) et des services web et qui proposent des algorithme de composition de services selon une fonction-objectif. Ces travaux peuvent être adaptés aux CM afin de fournir un raisonneur qui jouera le rôle d'un moteur de recherche performant. Aussi, le mécanisme des moteurs de recherche existants (google, yahoo, duckduckgo) peuvent être étudiés et explorés afin de fournir de modèle pour l'implémentation du raisonneur. Par exemple, [Bruno 12] proposent un modèle ontologique en implémentant un ensemble de règles pour faciliter la gestion des contrats dans un réseau formé de PME. D'autre part, la sécurité des données hébergées est au cœur des inquiétudes des clients et des fournisseurs. Les clients sont préoccupés par la confidentialité de leurs données, la sauvegarde et la restauration de ces dernières. Les fournisseurs doivent de leur coté, assurer une isolation des VMs, la mise en place de procédures de sauvegarde, de récupération et de protection de données. [Främling 07] expliquent que l'un des défis dans des systèmes en réseaux est de savoir où sont réellement les données, comment y accéder, et comment les mettre à jour. Ils expliquent qu'une manière de faire peut être l'attribution d'un identifiant unique permettant de trouver et d'identifier le produit concerné. Il existe dans la littérature des travaux et

études traitant la sécurité dans le CC qui peuvent être appliqués au CM afin d'assurer la protection des données. Il est aussi important que l'entreprise utilisant les services *cloud* dispose d'une grille de droits d'accès indiquant quels sont les utilisateurs autorisés à manipuler ses données dans le *cloud*. Par exemple [Goncalves 08], propose une méthodologie permettant de réaliser le mécanisme RBAC (Role-Based Access Control) durant la conception d'un SI afin de garantir et gérer la sécurité et les droits d'accès au sein d'un entreprise. En effet, cette proposition peut servir comme référence pour la gestion de la sécurité d'un SIE destiné à être migré vers le *cloud*.

6.9.2 Application sur un cas réel

Comme expliqué précédemment, nous avons effectué des tests unitaires afin de valider notre proposition. Bien que les résultats soient positifs, l'application sur un scénario industriel réel permet une validation définitive à travers l'évaluation et le test de la plate-forme dans des environnements comportant des milliers d'instances. En effet, les systèmes et environnements basés sur le cloud contiennent un nombre considérable d'entités fortement inter-connectées. Ces fortes inter-connexions peuvent être illustrées par le fait qu'un seul utilisateur est connecté aux différents fournisseurs par l'intermédiaire d'un ensemble de services qui, eux même sont décrits via les informations rentrées dans le système par les acteurs, et celles inférées. Devant ce volume croissant d'informations, la validation de notre proposition implique son évaluation par rapport à un ensemble de scénario divers de grandes quantités de scénarios, ce qui n'est pas une tâche facile. Dans le cadre de projets liés à la recherche et au développement que nous menons, nous effectuons des travaux afin de tester notre proposition et la valider sur des scénarios de clients industriels.

6.9.3 Les licences et droits d'utilisation

Une entreprise doit savoir de quels logiciels elle dispose, comment ils sont utilisés et de quelle manière seront ils impactés par la migration vers le cloud. Il s'agit de la gestion des actifs logiciels qui est définie comme étant « toutes les infrastructures et les processus nécessaires pour une gestion efficace, un contrôle et une protection des actifs logiciels de l'organisation et ce à toutes les étapes de leur cycle de vie »[6]. Le software Asset Management est une norme ISO. Le Software Asset Management doit permettre aux entreprises de gérer leurs actifs au quotidien et d'instaurer un bon équilibre, comme faire un inventaire, avoir une parfaite connaissance et une politique stricte des différents accès donnés aux employés et stagiaires, intégrer une bonne stratégie, fournir des autorisations en cadrant, et favoriser l'épanouissement de la productivité des salariés tout en contrôlant le SI et évitant toute vulnérabilité[7]. Mais ceci, s'applique aussi aux fournisseurs et revendeurs de services, notamment dans le cas des logiciels ou la licence constitue un sujet de malentendu. En effet, pour certains éditeurs de logiciel, une licence en réseau (un nombre d'utilisateurs d'une entreprise est autorisé à utiliser le logiciel avec accès simultané) est forcément nominative et peut être acheté pour le compte d'une société. Ainsi, un fournisseur n'a pas le droit d'acheter une telle

6. http://www.samsource.com/index.php?q=SAMSource_Library/Introduction/samsource_ITIL-ISO-and-SAMsource.sam consulté en décembre 2015

7. http://www.itpro.fr/a/bsa-the-software-alliance-instaurer-confiance/ consulté en Décembre 2015

licence pour fournir une offre Saas pour des utilisateurs issus d'entreprises différentes. Le guide SAM a soulevé cette question en expliquant : « Les entreprises peuvent se trouver exposées si le fournisseur de services Cloud enfreint les droits de propriété intellectuelle d'un tiers en mettant en service la solution. L'utilisation non autorisée de comptes SaaS crée également des risques de conformité. Il peut s'agir d'accéder au service depuis des zones prohibées, de partager des comptes utilisateurs, de permettre à des systèmes de se faire passer pour des utilisateurs ou d'autoriser des personnes autres que les employés (sous-traitants, fournisseurs ou clients) à accéder à des emplacements qui leur sont interdits. » Par conséquent, les guide des meilleures pratiques dans ce domaine doivent prendre en compte la protection contre la contrefaçon des logiciels. Ce point est ainsi primordial à la mise en place d'une plate-forme CM.

6.9.4 Vers le Green Cloud manufacturing

Selon une étude menée par James Glanz « les data-center du monde entier a consommé 30 milliards de watts en énergie en 2012, qui est équivalent à l'énergie produite par 30 centrales nucléaires [Ekman 15]. Il est ainsi impératif de trouver de nouvelles façons et méthodes afin d'optimiser les besoin en énergie de ces data-centers. La consommation électrique est à l'origine de beaucoup de problèmes dûs à la chaleur excessive. Les fournisseurs de services dans le domine du CC se focalisent aujourd'hui sur les méthodes d'isolations de VM et de la confidentialité des données et délaissent généralement la gestion de leur consommation électrique. Les fournisseurs de services doivent prendre des mesures et veiller à ce que leur marge bénéficiaire n'est pas réduite en raison des coûts de consommations électrique élevés. Il faut prendre en compte aussi la pression croissante des gouvernements incitant à réduire l'empreinte carbone[8] qui a un impact significatif sur le changement climatique. Des travaux ont été initiés par l'école partenaire ECAM-EPMI avec la même méthodologie utilisée dans cette thèse (ASCI) [Huet 13a].

8. Selon le dictionnaire de l'environnement, on appelle « empreinte carbone » la mesure du volume de dioxyde de carbone (CO_2) émis par combustion d'énergies fossiles, par les entreprises ou les êtres vivants.

7

ANNEXE

Complément d'information sur les classes de la *Cloud Manufacturing Ontology* (CMO)

Nous présentons dans cette partie les détails des classes qui composent la CMO

7.1 Classe **Actor**

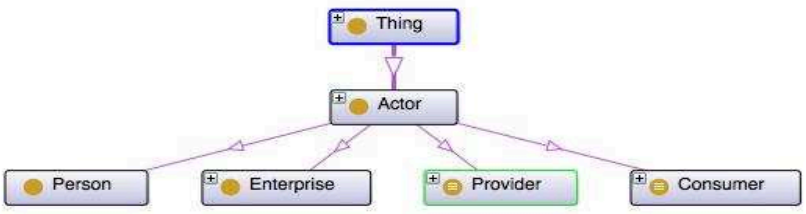


Figure 7.1 – *Détails de la classe **Actor***

7.2 Classe **CutomerFeedback**

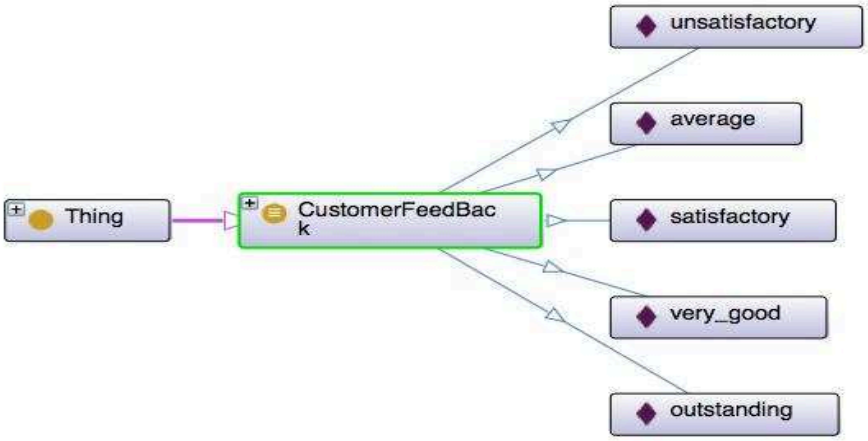


Figure 7.2 – *Détails de la classe* ***CutomerFeedback***

7.3 Classe **DeliveryMethod**

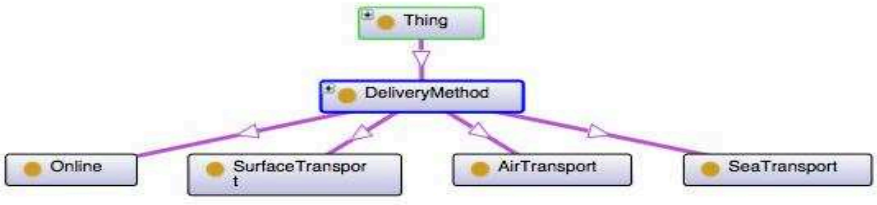


Figure 7.3 – *Détails de la classe* ***DeliveryMethod***

7.4 Classe **DeploymentModel**

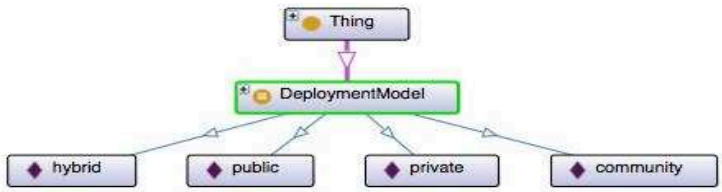


Figure 7.4 – *Détails de la classe* ***DeploymentModel***

7.5 Classe **PaymentMethod**

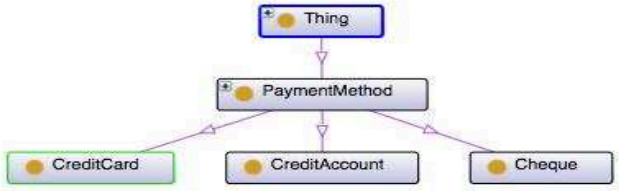


Figure 7.5 – *Détails de la classe* ***PaymentMethod***

7.6 Classe **Resource**

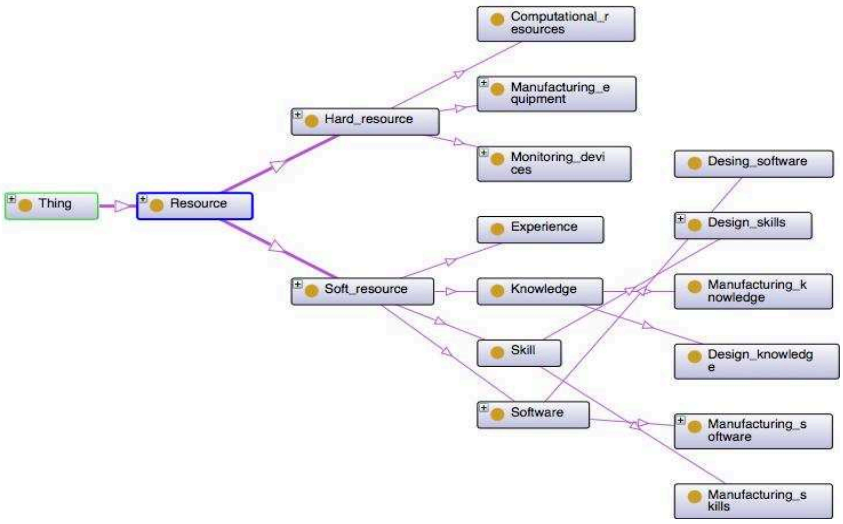


Figure 7.6 – *Détails de la classe* ***Resource***

7.7 Classe **Security**

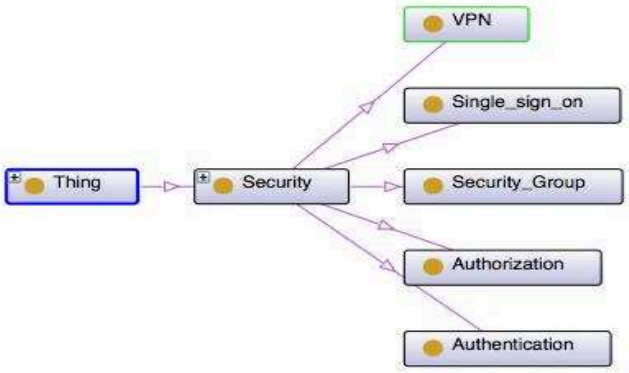


Figure 7.7 – *Détails de la classe* ***Security***

7.8 Classe **Semantic_elements**

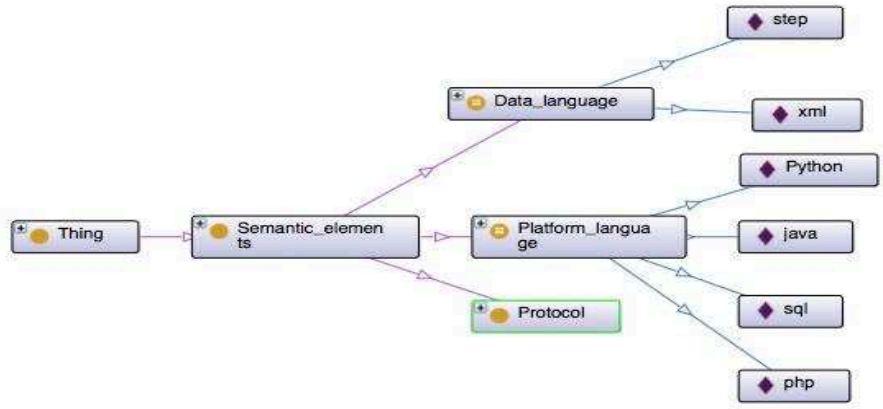


Figure 7.8 – *Détails de la classe* ***Semantic_elements***

7.9 Classe **Service**

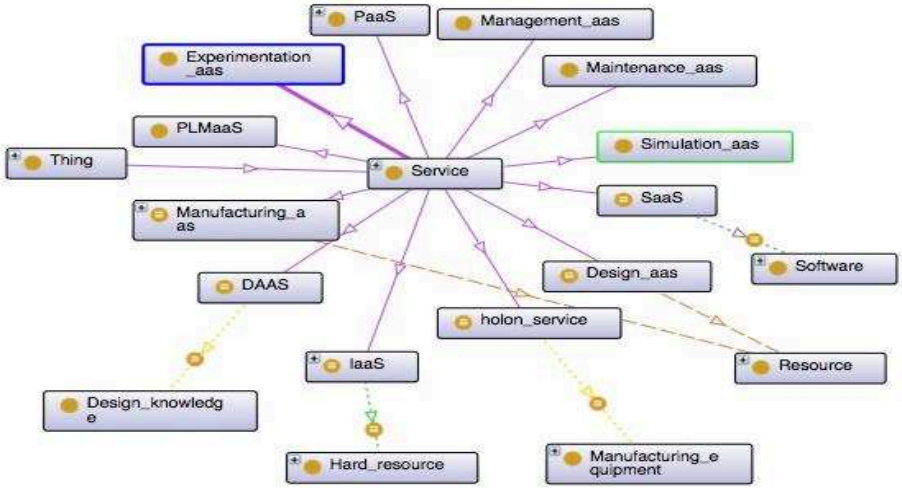


Figure 7.9 – *Détails de la classe* ***Service***

7.10 Classe **SLA**

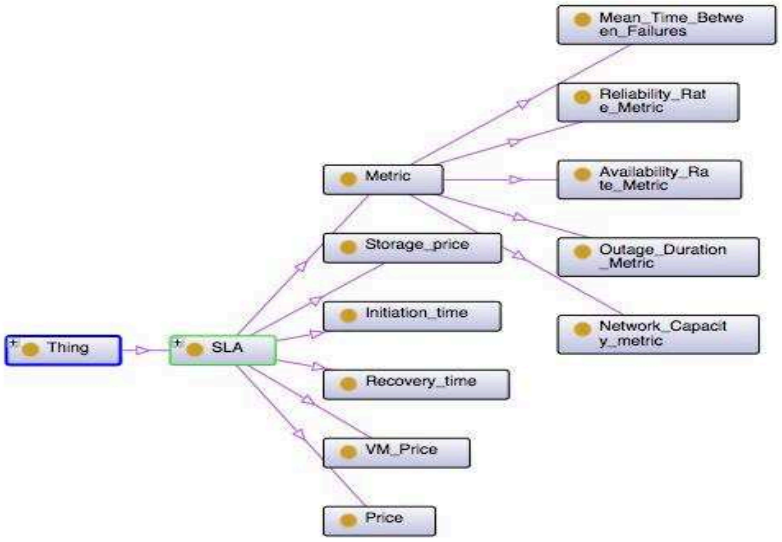


Figure 7.10 – *Détails de la classe* ***SLA***

7.11 Classe **State**

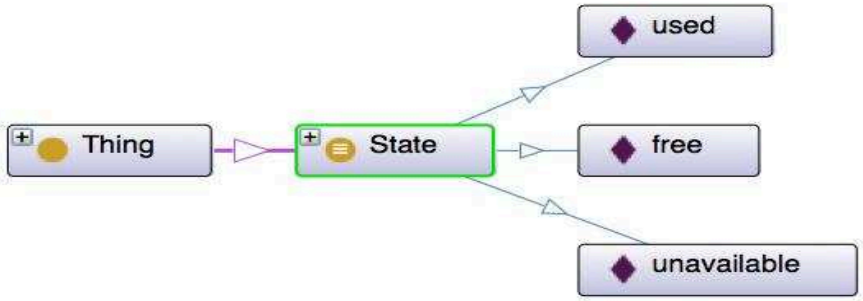


Figure 7.11 – *Détails de la classe* ***State***

Bibliographie

[Aberdeen_Group 07] Aberdeen_Group. *The Configuration Management Benchmark Report : Formalization and extending CM to drive qualilty*, 2007.

[Alberts 03] Christopher J. Alberts, Audrey J. Dorofee, Julia H. Allen, Christopher J. Alberts, Audrey J. Dorofee, Julia H. Allen, Norton L. Compton & Lt Col. *Power to the edge : Command*. In control.., in the information age. Washington : Command Control Research Program, 2003.

[Ali 09] Ahmad Ali. *L' approche multi-agents pour le pilotage des systèmes complexes appliquée aux systèmes du trafic urbain*. Theses, Université Blaise Pascal - Clermont-Ferrand II, July 2009.

[Alonso 04] Gustavo Alonso, Fabio Casati, Harumi Kuno & Vijay Machiraju. Web services. Springer, 2004.

[Ameri 12] Farhad Ameri, Colin Urbanovsky & Christian McArthur. *A Systematic Approach to Developing Ontologies for Manufacturing Service Modeling*. 7th International Conference on Formal Ontology in Information Systems (FOIS), Graz, Austria, 2012.

[Antoniou 04] Grigoris Antoniou & Frank Van Harmelen. *Web Ontology Language : OWL*. In Handbook on ontologies, pages 67–92. Springer, 2004.

[Armonk 07] N.Y. Armonk. *IBM Introduces Ready-to-Use Cloud Computing*, 2007.

[Baader 91] Franz Baader & Bernhard Hollunder. *A terminological knowledge representation system with complete inference algorithms*. In Processing Declarative Knowledge, pages 67–86. Springer, 1991.

[Baglin 15] Gérard Baglin, Samir Lamouri & André Thomas. Maîtriser les progiciels ERP. Editions Economica, March 2015.

[Baïna 06] Salah Baïna & Gérard Morel. *Product centric holons for synchronisation and interoperability in manufacturing environments*. In IFAC, editeur, 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'2006, St-Etienne, France, May 2006. IFAC.

[Baker 02] Mark Baker, Rajkumar Buyya & Domenico Laforenza. *Grids and Grid technologies for wide-area distributed computing*. Software : Practice and Experience, vol. 32, no. 15, pages 1437–1466, December 2002.

[Barkmeyer 99] Edward Barkmeyer, Peter Denno, Shaw Feng, Al Jones & Evan Wallace. *NIST response to MES request for information*. 1999.

[Bechhofer 01] Sean Bechhofer, Ian Horrocks, Carole Goble & Robert Stevens. *OilEd : a reason-able ontology editor for the semantic web*. In Ki 2001 : Advances in Artificial Intelligence, pages 396–408. Springer, 2001.

[Ben Naylor 99] J. Ben Naylor, Mohamed M. Naim & Danny Berry. *Leagility : integrating the lean and agile manufacturing paradigms in the total supply chain*. International Journal of Production Economics, vol. 62, no. 1, pages 107–118, 1999.

[Berners-Lee 01] Tim Berners-Lee, James Hendler & Ora Lassila. *The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. Scientific American, vol. 284, no. 5, pages 1–5, 2001.

[Berners-Lee 04] Tim Berners-Lee, Roy Fielding & Larry Masinter. *Uniform resource identifier (URI) : Generic syntax*, 2004.

[Bernus 97] Peter Bernus & Laszlo Nemes. *Requirements of the generic enterprise reference architecture and methodology*. Annual Reviews in Control, vol. 21, no. 3, pages 125–136, July 1997.

[Bezemer 10] Cor-Paul Bezemer & Andy Zaidman. *Multi-tenant SaaS applications : maintenance dream or nightmare ?* In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), pages 88–92. ACM, 2010.

[Borst 97] Pim Borst & Hans Akkermans. *An ontology approach to product disassembly*. In Knowledge Acquisition, Modeling and Management, pages 33–48. Springer, 1997.

[Bradshaw 10] Simon Bradshaw, Christopher Millard & Ian Walden. *Contracts for Clouds : Comparison and Analysis of the Terms and Conditions of Cloud Computing Services*. Legal Studies, vol. 19, no. 63, pages 1–47, 2010.

[Brahim-Djelloul 12] Sakina Brahim-Djelloul, Dominique Estampe & Samir Lamouri. *Real-time information management in Supply Chain modelling tools*. International Journal of Services Operations and Informatics, vol. 7, no. 4, pages 294–312, 2012.

[Brandis 14] Knud Brandis, Srdan Dzombeta & Knut Haufe. *Towards a framework for governance architecture management in cloud environments : A semantic perspective*. Future Generation Computer Systems, vol. 32, pages 274–281, March 2014.

[Brecher 09] C. Brecher, W. Lohse & M. Vitr. *Module-based platform for seamless interoperable CAD-CAM-CNC planning*. In Advanced design and manufacturing based on STEP, pages 439–462. Springer, 2009.

[Bresciani 04] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia & John Mylopoulos. *Tropos : An agent-oriented software development methodology*. Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, pages 203–236, 2004.

[Brown 96] Alan W Brown & Kurt C Wallnan. *Engineering of component-based systems*. In Engineering of Complex Computer Systems, 1996.

Proceedings., Second IEEE International Conference on, pages 414–422. IEEE, 1996.

[Browne 95] J. Browne, P.J. Sackett & J.C. Wortmann. *Future manufacturing systems–Towards the extended enterprise*. Computers in Industry, vol. 25, no. 3, pages 235–254, 1995.

[Bruno 12] Giulia Bruno & Agostino Villa. *An Ontology-Based Model for SME Network Contracts*. In Pilar Herrero, Hervé Panetto, Robert Meersman & Tharam Dillon, editeurs, On the Move to Meaningful Internet Systems : OTM 2012 Workshops, volume 7567 of *Lecture Notes in Computer Science*, pages 85–92. Springer Berlin Heidelberg, 2012.

[Brussel 98] Hendrik Van Brussel, Jo Wyns, Paul Valckenaers, Luc Bongaerts & Patrick Peeters. *Reference Architecture for Holonic Manufacturing Systems : PROSA*. Computers in Industry, vol. 37, no. 3, pages 255–276, 1998.

[Bussmann 99] S. Bussmann & D.C. McFarlane. *Rationales for holonic manufacturing control*. In Proc. of Second Int. Workshop on Intelligent Manufacturing Systems, pages 177–184, 1999.

[Buyya 02] Rajkumar Buyya & Manzur Murshed. *GridSim : a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing*. Concurrency and Computation : Practice and Experience, vol. 14, no. 13-15, pages 1175–1220, 2002.

[Buyya 09a] Rajkumar Buyya, Rajiv Ranjan & Rodrigo N Calheiros. *Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit : Challenges and opportunities*. In High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, pages 1–11. IEEE, 2009.

[Buyya 09b] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg & Ivona Brandic. *Cloud computing and emerging IT platforms : Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Computer Systems, vol. 25, no. 6, pages 599–616, June 2009.

[Calheiros 11] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose & Rajkumar Buyya. *CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*. Software : Practice and Experience, vol. 41, no. 1, pages 23–50, 2011.

[Campos 09] Julio Garrido Campos & Luis Rodriguez Miguez. *Manufacturing traceability data management in the supply chain*. International Journal of Information Technology and Management, vol. 8, no. 3, pages 321–339, 2009.

[Cardin 07] Olivier Cardin. *Apport de la simulation en ligne dans l'aide à la décision pour le pilotage des systèmes de production : Application à un système flexible de production*. Thèse de doctorat, Université de Nantes, 2007.

[Carr 06] Nicholas. Carr. *Rough type. Available online at : http ://www.roughtype.com/ ?p=279*. 2006.

[Castro 99] Miguel Castro, Barbara Liskov*et al. Practical Byzantine fault tolerance*. In Proceedings of the third symposium on Operating systems design and implementation, pages 173–186, 1999.

[Cavoukian 08] A Cavoukian. *Privacy in the Clouds : A White Paper on Privacy and Digital Identity–Implications for the Internet. May 28, 2008*, 2008.

[CC2] *White Paper - Cloud Computing II. The world of IT is changing*.

[CCW 10] *The Open Group : Building return on investment from cloud computing. A white paper, cloud business artifacts project. Cloud Computing Work Group*, 2010.

[Cea Ramirez 06] Aldo Cea Ramirez. *Contribution à la Modélisation et à la Gestion des Interactions Produit-Processus dans la Chaîne Logistique par l ?Approche Produits Communicants*. Theses, Université Henri Poincaré - Nancy I, July 2006.

[Cernuzzi 06] Luca Cernuzzi & Franco Zambonelli. *Dealing with adaptive multi-agent organizations in the gaia methodology*. In Agent-Oriented Software Engineering VI, pages 109–123. Springer, 2006.

[Chabrol 06] Michelle Chabrol, David Sarramia & Nikolay Tchernev. *Urban traffic systems modelling methodology*. International Journal of Production Economics, vol. 99, pages 156 – 176, 2006.

[Chabrol 08] M. Chabrol, M. Gourgand & S. Rodier. *A modeling methodology and its application to the design of decision-making aid tools dedicated to the hospital systems*. In RCIS'08 :International Conference on Research Challenges in Information, 2008.

[Chang 12] Hyokyung Chang, Hyosik Ahn & Euiin Choi. *Efficient Context Modeling Using OWL in Mobile Cloud Computing*. Energy Procedia, vol. 16, pages 1312–1317, January 2012.

[Charlet 02] Jean Charlet. *Knowledge Engineering : developments, results and perspectives for Knowledge Management in medical field*. Habilitation à diriger des recherches, Université Pierre et Marie Curie - Paris VI, December 2002.

[Chauvet 09] Julie Chauvet. *Une méthodologie de modélisation pour les systèmes hospitaliers : application sur le Nouvel Hôpital d'Estaing*. PhD thesis, Thèse de doctorat, Clermont-Ferrand, 2009.

[Chen 13] XiaoJun Chen, Jing Zhang, Junhuai Li & Xiang Li. *Resource virtualization methodology for on-demand allocation in cloud computing systems*. Service Oriented Computing and Applications, vol. 7, no. 2, pages 77–100, 2013.

[Cheng 10] Y Cheng, F Tao, L Zhang, X Zhang, GH Xi & D Zhao. *Study on the utility model and utility equilibrium of resource service transaction in cloud manufacturing*. In Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on, pages 2298–2302. IEEE, 2010.

[Chirn 00] JL Chirn & DC McFarlane. *A holonic component-based approach to reconfigurable manufacturing control architecture*. Database and Expert Systems, 2000.

[Chiron 08] Fabien Chiron. *Contribution à la flexibilité et à la rapidité de conception des systèmes automatisés avec l'utilisation d'UML*. Theses, Université Blaise Pascal - Clermont-Ferrand II, December 2008.

[Choudhary 07] V. Choudhary. *Software as a Service : Implications for Investment in Software Development*. In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on, pages 209a–209a, Jan 2007.

[Christopher 00] Martin Christopher & Denis R. Towill. *Supply chain migration from lean and functional to agile and customised*. Supply Chain Management : An International Journal, vol. 5, no. 4, pages 206–213, 2000.

[CNSS 06] CNSS. *NATIONAL INFORATION ASSURANCE (IA) GLOSSARY*, 2006.

[Cohen 08] Reuven Cohen. *Virtual Private Cloud (VPC)*. 2008.

[Conklin 87] Jeff Conklin. *Hypertext : An Introduction and Survey*. Computer, vol. 20, no. 9, pages 17–41, September 1987.

[Cutkosky 93] M.R. Cutkosky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W.S. Mark, J.M. Tenenbaum & J.C. Weber. *PACT : An experiment in integrating concurrent engineering systems*. Computer, vol. 26, no. 1, pages 28–37, 1993.

[Cutkosky 98] Mark Cutkosky, Larry Leifer, Charles Petrie & George Toye. *SHARE : A methodology and environment for collaborative product development*. Rapport technique, DTIC Document, 1998.

[David 09] Robbins David. *Cloud Computing Explained*, 2009.

[Dean 08] Jeffrey Dean & Sanjay Ghemawat. *MapReduce : simplified data processing on large clusters*. Communications of the ACM, vol. 51, no. 1, pages 107–113, 2008.

[Dillon 10a] Tharam Dillon, Chen Wu & Elizabeth Chang. *Cloud Computing : Issues and Challenges*. 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pages 27–33, 2010.

[Dillon 10b] Tharam Dillon, Chen Wu & Elizabeth Chang. *Cloud Computing : Issues and Challenges*. 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pages 27–33, 2010.

[Dukaric 13] Robert Dukaric & Matjaz B. Juric. *Towards a unified taxonomy and architecture of cloud frameworks*. Future Generation Computer Systems, vol. 29, no. 5, pages 1196–1210, July 2013.

[Dutta 05] D. Dutta & J.P. Wolowicz. *An introduction to Product Lifecycle Management (PLM)*. In ISPE, 2005.

[EIF 04] EIF. *European interoperability framework - White paper*, 2004.

[Eiter 08] Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner & Axel Polleres. *Rules and Ontologies for the Semantic Web*. In Cristina Baroglio, PieroA. Bonatti, Jan Małuszyński, Massimo Marchiori, Axel Polleres & Sebastian Schaffert, editeurs, Reasoning Web, volume 5224 of *Lecture Notes in Computer Science*, pages 1–53. Springer Berlin Heidelberg, 2008.

[Ekman 15] U. Ekman, J.D. Bolter, L. Diaz, M. Sondergaard & M. Engberg. Ubiquitous computing, complexity and culture. Taylor & Francis, 2015.

[El Haouzi 08a] H. El Haouzi, A. Thomas & J.-F. Pétin. *Contribution to reusability and modularity of manufacturing systems simulation models : Application to distributed control simulation within DFT context*. International Journal Production Economics, vol. 112, pages 48–61, 2008.

[El Haouzi 08b] Hind El Haouzi. *Approche méthodologique pour l'intégration des systèmes contrôlés par le produit dans un environnement de juste-à-temps : Application à l'entreprise Trane*. Theses, Université Henri Poincaré - Nancy I, November 2008.

[El Kadiri 09] Pernelle P. Delattre M. Bouras Abdelaziz El Kadiri Soumaya. *Current situation of PLM systems in SME/SMI : Survey's results and analysis*. In In : 6th International Conference on Product Lifecycle Management, Bath, UK, 2009.

[Elkadiri 08] S. Elkadiri, P. Pernelle, M. Delattre & A. Bouras. *Pilotage des processus collaboratifs dans les systèmes PLM. Quels indicateurs pour quelle évaluation des performances ?* In Actes du 1er Congrès des innovations mécaniques CIM'08, page 18 pages, Sousse, Tunisie, April 2008.

[F. Tao 10] L. Zhang F. Tao YF. Hu. Theory and practice : optimal resource service allocation in manufacturing grid. Bejing : China Machine Press, 2010.

[Fan 03] Yushun. Fan, Feil. Liu & Guoning Qi. *Network manufacturing system and its application*, 2003.

[Fan 04] Yushun Fan, Dazhe Zhao, Liqin Zhang, Shuangxi Huang & Bo Liu. *Manufacturing grid : needs, concept, and architecture*. Grid and Cooperative, pages 2–5, 2004.

[Fankam 08] Chimene Fankam. *OntoDB2 : Support of Multiple Ontology Models Within Ontology Based Database*. In Proceedings of the 2008 EDBT Ph.D. Workshop, Ph.D. '08, pages 21–27, New York, NY, USA, 2008. ACM.

[Fernández-López 97] Mariano Fernández-López, Asunción Gómez-Pérez & Natalia Juristo. *Methontology : from ontological art towards ontological engineering*. 1997.

[Fielding 00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[Fiorentini 10] Xenia Fiorentini, Sudarsan Rachuri, Hyowon Suh, Jaehyun Lee & Ram D Sriram. *An analysis of description logic augmented with domain rules for the development of product models*. Journal of computing and information science in engineering, vol. 10, no. 2, page 021008, 2010.

[Fortineau 13a] Virginie Fortineau. *Contribution à une modélisation ontologique des informations tout au long du cycle de vie du produit*. Theses, Ecole nationale supérieure d'arts et métiers - ENSAM, November 2013.

[Fortineau 13b] Virginie Fortineau, Thomas Paviot & Samir Lamouri. *5 root concepts for a meta-ontology to model product along its whole lifecycle*. 11th IFAC Workshop on Intelligent Manufacturing Systems, São Paulo, Brazil, 2013.

[Fortineau 13c] Virginie Fortineau, Thomas Paviot & Samir Lamouri. *Improving the interoperability of industrial information systems with description logic-based models ?The state of the art*. Computers in Industry, vol. 64, no. 4, pages 363–375, May 2013.

[Fortineau 14] Virginie Fortineau, Xenia Fiorentini, Thomas Paviot, Ludovic Louis-Sidney & Samir Lamouri. *Expressing formal rules within ontology–based models using SWRL : an application to the nuclear industry*. International Journal of Product Lifecycle Management, vol. 7, no. 1, pages 75–93, 2014.

[Foster 99] Ian Foster & Carl Kesselman, editeurs. The grid : Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[Främling 07] Kary Främling, Timo Ala-Risku, Mikko Kärkkäinen & Jan Holmström. *Design Patterns for Managing Product Life Cycle Information*. Commun. ACM, vol. 50, no. 6, pages 75–79, June 2007.

[Fuchs 08] Christian Fuchs. *Don Tapscott & Anthony D. Williams : Wikinomics : How Mass Collaboration Changes Everything*. International Journal of Communication, vol. 2, page 11, 2008.

[Fuh 05] Jerry YH Fuh & WD Li. *Advances in collaborative CAD : the-state-of-the art*. Computer-Aided Design, vol. 37, no. 5, pages 571–581, 2005.

[Funika 13] Wlodzimierz Funika, Michal Janczykowski, Konrad Jopek & Maciej Grzegorczyk. *An Ontology-based Approach to Performance Monitoring of MUSCLE-bound Multi-scale Applications*. Procedia Computer Science, vol. 18, pages 1126–1135, January 2013.

[Féniès 06] Pierre Féniès, Michel Gourgand & Sophie Rodier. *Interoperable and Multi-flow Software Environment : Application to Health Care Supply Chain*. In Johann Eder & Schahram Dustdar, editeurs, Business Process Management Workshops, volume 4103 of *Lecture Notes in Computer Science*, pages 311–322. Springer Berlin Heidelberg, 2006.

[Galland 03] S Galland, F Grimaud, P Beaune & J.P Campagne. *A- : An introduction to a methodological approach for the simulation of distributed*

industrial systems. International Journal of Production Economics, vol. 85, no. 1, pages 11–31, 2003.

[Galuba 09] Wojciech Galuba & Sarunas Girdzijauskas. *Distributed Hash Table*. In LING LIU & M.TAMER ÖZSU, editeurs, Encyclopedia of Database Systems, pages 903–904. Springer US, 2009.

[Gartner 09] Gartner. *Gartner Research : Predicts 2009 : Cloud Computing Beckons*, 2009.

[Geelan 09] Jeremy Geelan. *Twenty-One Experts Define Cloud Computing. Virtualization.*, 2009.

[Genin 05a] Patrick Genin, Samir LAMOURI & André THOMAS. *Planification avancée : APS*. Techniques de l'ingénieur. L'Entreprise industrielle, no. AG5120, 2005.

[Genin 05b] Patrick. Genin, Samir. Lamouri & André. Thomas. *La planification industrielle et ses limites*. pages 1–13, 2005.

[Germain 13] Jack M Germain. *How to Avoid Cloud Vendor Lock-In*, 2013.

[Giorgini 04] Paolo Giorgini, Manuel Kolp, John Mylopoulos & Marco Pistore. *The Tropos Methodology : An Overview*. Methodologies And Software Engineering For Agent Systems, pages 89–106, 2004.

[Giunchiglia 03] Fausto Giunchiglia, Fausto Giunchiglia, John Mylopoulos, John Mylopoulos, Anna Perini & Anna Perini. *The Tropos Software Development Methodology : Processes, Models and Diagrams*. Agent-Oriented Software Engineering III, vol. 2585, no. November 2001, pages 162–173, 2003.

[Golden 08] Bernard Golden. Virtualization for dummies. Wiley Publishing, Inc, 2008.

[Goncalves 08] Gilles Goncalves & Aneta Poniszewska-Maranda. *Role engineering : From design to evolution of security schemes*. Journal of Systems and Software, vol. 81, no. 8, pages 1306 – 1326, 2008.

[Gong 10] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen & Zhenghu Gong. *The characteristics of cloud computing*. Proceedings of the International Conference on Parallel Processing Workshops, pages 275–279, 2010.

[Gottschalk 00] Klaus Gottschalk. *Web services architecture overview. IBM white paper*, 2000.

[Gourgand 91] Michel Gourgand & Patric Kellert. *Conception d'un environnement de modélisation des systèmes de production*. In 3ème congrès international de génie industriel, Tour, France, 1991.

[Gruber 95] Thomas R Gruber. *Toward principles for the design of ontologies used for knowledge sharing ?* International journal of human-computer studies, vol. 43, no. 5, pages 907–928, 1995.

[Guarino 98] Nicola Guarino. Formal ontology in information systems : Proceedings of the first international conference (fois'98), june 6-8, trento, italy, volume 46. IOS press, 1998.

[Guarino 00] Nicola Guarino & Christopher Welty. *Identity, unity, and individuality : Towards a formal toolkit for ontological analysis*. In ECAI, volume 2000, pages 219–223. Citeseer, 2000.

[Guarino 02] Nicola Guarino & Christopher Welty. *Identity and Subsumption*. In Rebecca Green, CarolA. Bean & SungHyon Myaeng, editeurs, The Semantics of Relationships, volume 3 of *Information Science and Knowledge Management*, pages 111–126. Springer Netherlands, 2002.

[Guarino 09] Nicola Guarino & Christopher A Welty. *An overview of OntoClean*. In Handbook on ontologies, pages 201–220. Springer, 2009.

[Guerra-Zubiaga 06] D. Guerra-Zubiaga, L. Donato, R. RamÃrez & M. Contero. *Knowledge Sharing to Support Collaborative Engineering at PLM Environment*. In Ulrich Reimer & Dimitris Karagiannis, editeurs, Practical Aspects of Knowledge Management, volume 4333 of *Lecture Notes in Computer Science*, pages 86–96. Springer Berlin Heidelberg, 2006.

[Guo 10] Hua Guo, Lin Zhang, Fei Tao, Lei Ren & Yong Liang Luo. *Research on the measurement method of flexibility of resource service composition in cloud manufacturing*. In Advanced Materials Research, volume 139, pages 1451–1454. Trans Tech Publ, 2010.

[Halpern 12] M. Halpern M.and Dominy, D. Scheibenreif & S. Jacobson. *A quick look at Cloud Computing in manufacturing industries, 2012*. Gartner, INC, pages 1–13, 2012.

[He 14] Wu He & Lida Xu. *A state-of-the-art survey of cloud manufacturing*. International Journal of Computer Integrated Manufacturing, no. May, pages 1–12, February 2014.

[Hernandez 05] Nathalie Hernandez. *Ontologies de domaine pour la modélisation du contexte en Recherche d'information*. Theses, Université Paul Sabatier - Toulouse III, December 2005.

[Horrocks 02] Ian Horrocks*et al. DAML+OIL : A Description Logic for the Semantic Web*. IEEE Data Eng. Bull., vol. 25, no. 1, pages 4–9, 2002.

[Horrocks 04] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean*et al. SWRL : A semantic web rule language combining OWL and RuleML*. W3C Member submission, vol. 21, page 79, 2004.

[Huang 07] George Q. Huang, Y.F. Zhang & P.Y. Jiang. *RFID-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts*. Robotics and Computer-Integrated Manufacturing, vol. 23, no. 4, pages 469 – 477, 2007.

[Huang 09] GQ Huang, PK Wright & Stephen T Newman. *Wireless manufacturing : a literature review, recent developments, and case studies*. International Journal of Computer Integrated Manufacturing, vol. 22, no. 7, pages 579–594, 2009.

[Huet 11] Jean-Charles Huet. *Proposition d'une méthodologie de réingénierie pour le contrôle par le produit de systèmes manufacturiers : application*

au circuit du médicament d ?un hôpital. Theses, Université Blaise Pascal - Clermont-Ferrand II, January 2011.

[Huet 13a] Jean-Charles Huet & Ikram El Abbassi. *Green Cloud Computing Modelling Methodology*. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC '13, pages 339–344, Dresden, Germany, 2013. IEEE Computer Society.

[Huet 13b] Jean-Charles Huet, Jean-Luc Paris, Khalid Kouiss & Michel Gourgand. *A new reengineering methodology for the product-driven system applied to the medication-use process*. Decision Support Systems, vol. 55, no. 2, pages 599 – 615, 2013.

[Ibrahim 11] A.S. Ibrahim, J. Hamlyn-Harris, John Grundy & M. Almorsy. *CloudSec : A security monitoring appliance for Virtual Machines in the IaaS cloud model*. In Network and System Security (NSS), 2011 5th International Conference on, pages 113–120, Sept 2011.

[ISO-9241-210 10] ISO-9241-210. *Ergonomics of human-system interaction – Part 210 : Human-centred design for interactive systems*, 2010.

[Jagdev 01] S. Jagdev, H. & Thoben K.D. *Anatomy of enterprise collaborations*. Production Planning and Control, vol. 12, no. 5, pages 437–451, 2001.

[Janardanan 08] V.K. Janardanan, M. Adithan & P. Radhakrishnan. *Collaborative product structure management for assembly modeling*. Computers in Industry, vol. 59, no. 8, pages 820 – 832, 2008.

[Jansen 11] Wayne Jansen & Timothy Grance. *Guidelines on Security and Privacy in Public Cloud Computing*. Director, vol. 144, pages 800–144, 2011.

[Jiang 10] Yang Jiang, Gaoliang Peng & Wenjian Liu. *Research on ontology-based integration of product knowledge for collaborative manufacturing*. The International Journal of Advanced Manufacturing Technology, vol. 49, no. 9-12, pages 1209–1221, 2010.

[Kabmala 06] Malee Kabmala, Lampang Manmart & Chaiyaporn Chirathamjaree. *An Ontology Based Approach To The Integration Of Heterogeneous Information Systems Supporting Integrated Provincial Administration In Khon Kaen, Thailand*. Edith Cowan University, Western Australia in association with Khon Kaen University, Thailand and Bansomdejchaopraya Rajabhat University, Thailand., 2006.

[Kepes 13] Ben Kepes. *Understanding the Cloud Computing Stack : SaaS, PaaS, IaaS. White paper*, 2013.

[Kiko 06] Kilian Kiko & Colin Atkinson. *A Detailed Comparison of UML and OWL*, 2006.

[Kim 13] Myougnjin Kim, Hanku Lee, Hyogun Yoon, Jee-In Kim & Hyung-Seok Kim. *IMAV : an intelligent multi-agent model based on cloud computing for resource virtualization*. In 2011 International Conference on Information and Electronics Engineering, IPCSIT, volume 6, 2013.

[Kiritsis 11] Dimitris Kiritsis. *Closed-loop PLM for intelligent products in the era of the Internet of things*. Computer-Aided Design, vol. 43, no. 5, pages 479–501, 2011.

[Kleinrock 05] L. Kleinrock. *A Vision for the Internet*. ST Journal for Research, vol. 2, no. 1, pages 4–5, November 2005.

[Kliazovich 10] Dzmitry Kliazovich, Pascal Bouvry & Samee Ullah Khan. *GreenCloud : a packet-level simulator of energy-aware cloud computing data centers*. The Journal of Supercomputing, vol. 62, no. 3, pages 1263–1283, 2010.

[Koestler 89] A. Koestler. The ghost in the machine. An Arkana book : philosophy. Arkana, 1989.

[Kontchakov 10] Roman Kontchakov, Frank Wolter & Michael Zakharyaschev. *Logic-based ontology comparison and module extraction, with an application to DL-Lite*. Artificial Intelligence, vol. 174, no. 15, pages 1093–1141, 2010.

[Kourtesis 14] Dimitrios Kourtesis, Jose María Alvarez-Rodríguez & Iraklis Paraskakis. *Semantic-based QoS management in cloud systems : Current status and future challenges*. Future Generation Computer Systems, vol. 32, pages 307–323, March 2014.

[Krima 09] Sylvere Krima, Raphael Barbau, Xenia Fiorentini, Rachuri Sudarsan, S Foufou & RD Sriram. *Ontostep : Owl-dl ontology for step*. National Institue of Standards and Technology, NISTIR, vol. 7561, 2009.

[Kunesch 10] Uli Kunesch, Martin Reti & Michael Pauly. *Cloud Computing I. Alternative sourcing Strategy for Business ICT. White paper*, 2010.

[Laili 12] Y. Laili, F. Tao, L. Zhang & B.R Sarker. *A study of optimal allocation of computing resources in cloud manufacturing systems*. The International Journal of Advanced Manufacturing Technology, vol. 63, no. 5-8, pages 671–690, 2012.

[Lamouri 06] S Lamouri. *Synchronisation des prises de décisions dans une chaîne logistique : robustesse et stabilité*. Mémoire de HDR, Institut Supérieur de Mécanique de Paris, 2006.

[Lassila 99] O. Lassila & R. R. Swick. *Resource description framework (rdf) model and syntax specification w3c recommendation 22 février 1999*, 1999.

[Le Moigne 92] J.-L. Le Moigne. La modélisation des systèmes complexes. 1992.

[Leavitt 09] N. Leavitt. *Is Cloud Computing Really Ready for Prime Time ?* Computer, vol. 42, no. 1, pages 15–20, Jan 2009.

[Lee 08] S.G. Lee, Y.S. Ma, G.L. Thimm & J. Verstraeten. *Product lifecycle management in aviation maintenance, repair and overhaul*. Computers in Industry, vol. 59, no. 2â3, pages 296 – 303, 2008.

[Lee 09] Jeongsoo Lee, Heekwon Chae, Cheol-Han Kim & Kwangsoo Kim. *Design of product ontology architecture for collaborative enterprises*. Expert Systems with Applications, vol. 36, no. 2, pages 2300–2309, 2009.

[Lee 10] J.H. Lee, S.J. Fenves, C. Bock, Hyo-Won Suh, S. Rachuri, X. Fiorentini & R.D. Sriram. *Product modeling framework and language for behavior evaluation*. In Automation Science and Engineering (CASE), 2010 IEEE Conference on, pages 136–143, Aug 2010.

[Leitão 04] P Leitão. *An agile and adaptive holonic architecture for manufacturing control*. PhD thesis, University of Porto, Portugal, 2004.

[Lemieux 13] Andrée-Anne Lemieux, Robert Pellerin & Samir Lamouri. *A Mixed Performance and Adoption Alignment Framework for Guiding Leanness and Agility Improvement Initiatives in Product Development*. Journal of Enterprise Transformation, vol. 3, no. 3, pages 161–186, 2013.

[Li 10a] B.H. Li, L. Zhang, S.L. Wang, F. Tao, J.W. Cao, X.D. Jiang, X. Song & X.D. Chai. *Cloud manufacturing : a new service-oriented networked manufacturing model*. Computer Integrated Manufacturing Systems, vol. 16, no. 1, pages 1–7, 2010.

[Li 10b] Q. Li, J. Zhou, Q.R. Peng, C.Q. Li, C. Wang, J. Wu & B.E Shao. *Business processes oriented heterogeneous systems integration platform for networked enterprises*. Computers in Industry, vol. 61, no. 2, pages 127–144, February 2010.

[Lim 09] Seung-Hwan Lim, B. Sharma, Gunwoo Nam, Eun Kyoung Kim & C.R. Das. *MDCSim : A multi-tier data center simulation, platform*. In Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on, pages 1–9, Aug 2009.

[Lin 11] LF Lin, WY Zhang, YC Lou, CY Chu & M Cai. *Developing manufacturing ontologies for knowledge reuse in distributed manufacturing environment*. International Journal of Production Research, vol. 49, no. 2, pages 343–359, 2011.

[Lindberg 93] Donald A Lindberg, Betsy L Humphreys & Alexa T McCray. *The Unified Medical Language System.* Methods of information in medicine, vol. 32, no. 4, pages 281–291, 1993.

[Liu 08] Qiong Liu & YongJiang Shi. *Gird manufacturing : a new solution for cross-enterprise collaboration*. The International Journal of Advanced Manufacturing Technology, vol. 36, no. 1-2, pages 205–212, 2008.

[Liu 11] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger & Dawn Leaf. *NIST cloud computing reference architecture*. NIST special publication, vol. 500, page 292, 2011.

[Liu 12] Ning Liu & Xiaoping Li. *A resource virtualization mechanism for cloud manufacturing systems*. In Enterprise Interoperability, pages 46–59. Springer, 2012.

[Liu 14] Chi-Lun Liu. *Cloud service access control system based on ontologies*. Advances in Engineering Software, vol. 69, pages 26–36, March 2014.

[López 99] Mariano Fernández López, Asunción Gómez-Pérez, Juan Pazos Sierra & Alejandro Pazos Sierra. *Building a chemical ontology using*

methontology and the ontology design environment. IEEE intelligent Systems, vol. 14, no. 1, pages 37–46, 1999.

[Louth 09] W Louth. *Metering the cloud : applying activity based costing (ABC) from code profiling up to performance and cost management of cloud computing*. In Proceedings of the International conference on JAVA technology, 2009.

[Lu 14] Yuqian Lu, Xun Xu & Jenny Xu. *Development of a Hybrid Manufacturing Cloud*. Journal of Manufacturing Systems, May 2014.

[Lv 12] Beisheng Lv. *A Multi-view Model Study for the Architecture of Cloud Manufacturing*. 2012 Third International Conference on Digital Manufacturing & Automation, pages 93–97, July 2012.

[Maedche 01] Alexander Maedche & Steffen Staab. *Ontology Learning for the Semantic Web*. IEEE Intelligent Systems, vol. 16, no. 2, pages 72–79, March 2001.

[Mařík 02] Vladimir Mařík, Martyn Fletcher & Michal Pěchouček. *Holons & agents : Recent developments and mutual impacts*. In Multi-Agent Systems and Applications II, pages 233–267. Springer, 2002.

[Massuthe 05] Peter Massuthe, Wolfgang Reisig & Karsten Schmidt. *An operating guideline approach to the SOA*. 2005.

[Mayanka Katyal 13] Atul Mishra Mayanka Katyal. *A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment*. International Journal of Distributed and Cloud Computing, vol. 1, no. 2, page 14, 2013.

[McFarlane 00] Duncan C McFarlane & Stefan Bussmann. *Developments in holonic production planning and control*. Production Planning & Control, vol. 11, no. 6, pages 522–536, 2000.

[Mebrek 08] Fateh Mebrek. *Outils d'aide à la décision basés sur la simulation pour la logistique hospitalière, application à un nouvel hôpital*. Theses, Université Blaise Pascal - Clermont-Ferrand II, December 2008.

[Meier 10] Matthias Meier, Joachim Seidelmann & István Mezgár. *ManuCloud : The Next-Generation Manufacturing as a Service Environment.* ERCIM News, vol. 2010, no. 83, pages 33–34, 2010.

[Mell 09] Peter Mell & Tim Grance. *The NIST definition of cloud computing*. National Institute of Standards and Technology, vol. 53, no. 6, page 50, 2009.

[Meyer 09] Gerben G. Meyer, Kary Främling & Jan Holmström. *Intelligent Products : A survey*. Computers in Industry, vol. 60, no. 3, pages 137 – 148, 2009.

[Michaloski 09] John Michaloski, Byeongeon Lee, Frederick Proctor, Sid Venkatesh & Sidney Ly. *Quantifying the performance of MT-Connect in a distributed manufacturing environment*. In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pages 533–539. American Society of Mechanical Engineers, 2009.

[Miller 98] George A Miller. *Nouns in wordnet*. WordNet : An electronic lexical database, pages 24–45, 1998.

[Ming 07] X.G. Ming, J.Q. Yan, W.F. Lu, D.Z. Ma & B. Song. *Mass production of tooling product families via modular feature-based design to manufacturing collaboration in PLM*. Journal of Intelligent Manufacturing, vol. 18, no. 1, pages 185–195, April 2007.

[Mokhtar 10] A. Mokhtar & M. Houshmand. *Introducing a roadmap to implement the universal manufacturing platform using axiomatic design theory*. International Journal of Manufacturing Research, vol. 5, no. 2, pages 252–269, 2010.

[Molina 07] A. Molina, H. Panetto, D. Chen, L. Whitman, V. Chapurlat, F. Vernadat*et al. Enterprise integration and networking : challenges and trends*. Studies in Informatics and Control, vol. 16, no. 4, pages 353–368, 2007.

[Moscato 11] Francesco Moscato, Beniamino Di Martino & Victor Munteanu. *An Analysis of mOSAIC ontology for Cloud Resources annotation*. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, Szczecin, Poland,, pages 973–980, 2011.

[Nagorny 12] K. Nagorny, A.W. Colombo & U.W.E Schmidtmann. *A service- and multi-agent-oriented manufacturing automation architecture : An {IEC} 62264 level 2 compliant implementation*. Computers in Industry, vol. 63, no. 8, pages 813 – 823, 2012.

[Nfaoui 08] El Habib Nfaoui. *Architecture d'aide à la décision distribuée et de simulation proactive dans les chaînes logistiques : Une approche multi agent*. 2008.

[Nick 10] Jeffrey M Nick, David Cohen & Burton S Kaliski Jr. *Key enabling technologies for virtual private clouds*. In Handbook of Cloud Computing, pages 47–63. Springer, 2010.

[Numata 98] J. Numata & Y. Maeda. *Engineering management for knowledge amplification in new product development*. In Engineering and Technology Management, 1998. Pioneering New Technologies : Management Issues and Challenges in the Third Millennium. IEMC '98 Proceedings. International Conference on, pages 241–246, 1998.

[OASIS 04] OASIS. *: Security services technical committe ,Security Assertion Markup Language (SAML). 2.0 Technical Overview Working Draft*, 2004.

[OMG 03] OMG. Mda guide version 1.0.1, volume omg/2003-06-01. juin 2003.

[Ouzrout 12] Y. Ouzrout. *Architectures collaboratives pour l'aide à la décision dans les entreprises en réseau*. Hdr, Université Lumière - Lyon II, March 2012.

[Pannequin 07] Rémi Pannequin. *Proposition d'un environnement de modélisation et de test d'architectures de pilotage par le produit de systèmes de production*. Theses, Université Henri Poincaré - Nancy I, July 2007.

[Park 12] J. Park & H.Y. Jeong. *The QoS-based MCDM system for SaaS ERP applications with Social Network*. The Journal of Supercomputing, pages 1–19, 2012.

[Paviot 10] Thomas Paviot. *Méthodologie de résolution des problèmes d'interopérabilité dans le domaine du Product Lifecycle Management*. PhD thesis, École Centrale Paris, 2010.

[Paviot 11] Thomas Paviot, Samir Lamouri & Vincent Cheutet. *A generic MultiCAD/MultiPDM interoperability framework*. International Journal of Services Operations and Informatics, vol. 6, no. 1/2, pages 124–137, 2011.

[Pavón 06] Juan Pavón. *Ingenias : Développement dirigé par modèles des systèmes multi-agents*. Habilitationa diriger des recherches de l'Université Pierre et Marie Curie, 2006.

[Pfister 98] Gregory F. Pfister. In search of clusters (2nd ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[Pratt 05] Michael J. Pratt, Bill D. Anderson & Tony Ranger. *Towards the standardized exchange of parameterized feature-based {CAD} models*. Computer-Aided Design, vol. 37, no. 12, pages 1251 – 1265, 2005.

[Ren 11] Lei Ren, Lin Zhang, Ya-Bin Zhang, Fei Tao & Yong-Liang Luo. *Resource virtualization in cloud manufacturing*. Computer integrated manufacturing systems, vol. 17, no. 3, pages 511–518, 2011.

[Richards 06] Robert Richards. *Representational State Transfer (REST)*. Pro PHP XML and Web Services SE - 17, pages 633–672, 2006.

[Rimal 10] Bhaskar Prasad Rimal, Admela Jukan, Dimitrios Katsaros & Yves Goeleven. *Architectural Requirements for Cloud Computing Systems : An Enterprise Cloud Approach*. Journal of Grid Computing, vol. 9, no. 1, pages 3–26, 2010.

[Roche 12] Christophe Roche. *e Ontologie : entre terminologie et connaissance de spécialité*. 2012.

[Rodier 10] Sophie Rodier. *Une tentative d'unification et de résolution des problèmes de modélisation et d'optimisation dans les systèmes hospitaliers*. Theses, Université Blaise Pascal - Clermont-Ferrand II, July 2010.

[Rodríguez 05] K. Rodríguez & A. Al-Ashaab. *Knowledge web-based system architecture for collaborative product development*. Computers in Industry, vol. 56, no. 1, pages 125–140, 2005.

[Rosemann 99] Michael Rosemann. *ERP-software-characteristics and consequences*. In In : Proceeding of the 7th European Conference on Information Systems, ECIS'99, Copenhaven, DK, 1999.

[Rosén 10] Jonas Rosén. *Development of Industrial Information Systems based on Standards*. PhD thesis, Production Engineering, Royal Institute of Technology, KTH, Stockholm, 2010.

[Royer 14] Johan Royer, Michelle Chabrol & Jean-Luc Paris. *Process modelling and simulation for medication-use process*. In 28th EUROPEAN Conference on Modelling and Simulation (ECMS), page 7 pages, Brescia, Italy, May 2014.

[Sahoo 10] J. Sahoo, S. Mohapatra & R. Lath. *Virtualization : A Survey on Concepts, Taxonomy and Associated Security Issues*. In Computer and Network Technology (ICCNT), 2010 Second International Conference on, pages 222–226, April 2010.

[Schaefer 12] Dirk Schaefer, JL Thames, RD Wellman, D Wu, S Yim & D Rosen. *Distributed collaborative design and manufacture in the cloud ?motivation, infrastructure, and education*. In 2012 American Society for Engineering Education Annual Conference, San Antonio, Paper# AC2012-3017, 2012.

[Servat 05] David Servat, Fabien Chiron, Khalid Kouiss, Thomas Meurisse & Guy Souchet. *Modèle d'architecture générique pour la supervision de systèmes de production*. In IDM'05 - Journées sur l'Ingénierie Dirigée par les Modèles, 2005.

[Shawish 14] Ahmed Shawish & Maria Salama. *Cloud Computing : Paradigms and Technologies*. In Inter-cooperative Collective Intelligence : Techniques and Applications, volume 495, pages 39–68. Springer, 2014.

[Shi 07] SY Shi, Rong Mo, H-C Yang, Z-Y Chang & Z-F Chen. *An implementation of modelling resource in a manufacturing grid for resource sharing*. International Journal of Computer Integrated Manufacturing, vol. 20, no. 2-3, pages 169–177, 2007.

[Shor 11] RM Shor. Cloud computing for learning and performance professionals, volume 1104. American Society for Training and Development, 2011.

[sia 01] *Software Information Industry Association : Software as a service : Strategic backgrounder*. 2001.

[Simon 62] Herbert-A Simon. *The architecture of complexity*. In Proceedings of the American Philosophical Society, numéro 6, pages 106(6) :467–482, 1962.

[Stadtler 05] Hartmut Stadtler. *Supply chain management and advanced planning–basics, overview and challenges*. European journal of operational research, vol. 163, no. 3, pages 575–588, 2005.

[Staisch 12] A. Staisch, G. Peters, T. Stueckl & J. Sergua. *Current trends in product lifecycle management*. In ACIS 2012 : Location, location, location : Proceedings of the 23rd Australasian Conference on Information Systems 2012, pages 1–10. ACIS, 2012.

[Stark 11] John Stark. Product lifecycle management : 21st century paradigm for product realisation. Springer, London, 2 edition, 2011.

[Studer 98] Rudi Studer, V Richard Benjamins & Dieter Fensel. *Knowledge engineering : principles and methods*. Data & knowledge engineering, vol. 25, no. 1, pages 161–197, 1998.

[Suh 08] S.H. Suh, S.J. Shin, J.S. Yoon & J.M. Um. *UbiDM : A new paradigm for product design and manufacturing via ubiquitous computing technology*. International Journal of Computer Integrated Manufacturing, vol. 21, no. 5, pages 540–549, 2008.

[Sun 11] H. Sun, N. Wan, Z. Chang & R. Mo. *Approach to optimization of part machining service combination*. The International Journal of Advanced Manufacturing Technology, vol. 56, no. 5-8, pages 767–776, 2011.

[Talhi 13] Asma Talhi, Jean-Charles Huet, Virginie Fortineau, Samir Lamouri & Xavier Lopez. *Study of a new tool to overcome collaboration issues within PLM : the Cloud Manufacturing*. 8ème Conférence Internationale Conception Production Intégrées, CPI'13, Tlemcen, Algérie., 21–23 octobre 2013.

[Talhi 14] Asma Talhi, Jean-Charles Huet, Virginie Fortineau & Samir Lamouri. *Toward an Ontology-based Architecture for Cloud Manufacturing*. 4th Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing, SOHOMA'14, Nancy, France, pages 164–173, 2014.

[Talhi 15a] A. Talhi, J.C. Huet, V. Fortineau & S. Lamouri. *Proposition d'un modeèle générique de connaissance pour le Cloud Manufacturing*. In 6èmes Journées Doctorales / Journées Nationales (JD-JN) MACS, Bourges, France, 2015.

[Talhi 15b] Asma Talhi, Jean-Charles Huet, Virginie Fortineau & Samir Lamouri. *A new ontology-based methodology applied to the Cloud Manufacturing*. The 16th Asia pacific industrial engineering and management systems conference - APIEMS'15, 2015.

[Talhi 15c] Asma Talhi, Jean-Charles Huet, Virginie Fortineau & Samir Lamouri. *Towards a Cloud Manufacturing systems modeling methodology*. 15th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'15, Ottawa, Canada, 2015.

[Tao 07] Fei Tao, Ye Fa Hu & Zu De Zhou. *Study on manufacturing grid & its resource service optimal-selection system*. The International Journal of Advanced Manufacturing Technology, vol. 37, no. 9-10, pages 1022–1041, April 2007.

[Tao 08] Fei Tao, Dongming Zhao, Yefa Hu & Zude Zhou. *Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system*. Industrial Informatics, IEEE Transactions on, vol. 4, no. 4, pages 315–327, 2008.

[Tao 10] F Tao, YF Hu & L Zhang. *Theory and practice : optimal resource service allocation in manufacturing grid*. 2010.

[Tao 11] F. Tao, L. Zhang, V.C. Venkatesh, Y. Luo & Y. Cheng. *Cloud manufacturing : a computing and service-oriented manufacturing model*. Proceedings of the Institution of Mechanical Engineers, Part B : Journal of Engineering Manufacture, 2011.

[Terzi 05] S. Terzi. *Elements of Product Lifecycle Management : Definitions, Open Issues and Reference Models*. PhD thesis, Université Henri Poincaré, Nancy-I, 2005.

[Terzi 10] Sergio Terzi, Abdelaziz Bouras, Debashi Dutta, Marco Garetti & Dimitris Kiritsis. *Product lifecycle management-from its history to its*

new role. International Journal of Product Lifecycle Management, vol. 4, no. 4, pages 360–389, 2010.

[Trentesaux 15] D. Trentesaux, T. Knothe, G. Branger & K. Fischer. *Planning and Control of Maintenance, Repair and Overhaul Operations of a Fleet of Complex Transportation Systems : A Cyber-Physical System Approach*. In Theodor Borangiu, André Thomas & Damien Trentesaux, editeurs, Service Orientation in Holonic and Multi-agent Manufacturing, volume 594 of *Studies in Computational Intelligence*, pages 175–186. Springer International Publishing, 2015.

[Tricot 95] André Tricot. *Un point sur l'ergonomie des interfaces hypermédias*. Le travail humain, vol. 58, no. 1, pages 17–45, 1995.

[Tsai 08] W.T. Tsai, Xin Sun, Qian Huang & Helen Karatza. *An ontology-based collaborative service-oriented simulation framework with Microsoft Robotics Studio®*. Simulation Modelling Practice and Theory, vol. 16, no. 9, pages 1392–1414, October 2008.

[Tsuchiya 93] Shigehisa Tsuchiya. *Improving knowledge creation ability trough organizational learning*. In Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK, Compiègne, France, 1993.

[Turner 03] M. Turner, D. Budgen & P. Brereton. *Turning software into a service*. Computer, vol. 36, no. 10, pages 38–44, Oct 2003.

[Uschold 95] Michael Uschold & Martin King. Towards a methodology for building ontologies. Citeseer, 1995.

[Valilai 13] O.F. Valilai & M. Houshmand. *A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm*. Robotics and Computer-Integrated Manufacturing, vol. 29, no. 1, pages 110 – 127, 2013.

[Van Der Velde 09] P.J.M.C. Van Der Velde. *Runtime Configurable Systems for Computational Fluid Dynamics Simulations*. PhD thesis, Auckland : Department of Mechanical Engineering, University of Auckland, 2009.

[van Heijst 97] G. van Heijst, A.Th. Schreiber & B.J. Wielinga. *Using explicit ontologies in {KBS} development*. International Journal of Human-Computer Studies, vol. 46, no. 2â3, pages 183 – 292, 1997.

[Vichare 09] Parag Vichare, Aydin Nassehi, Sanjeev Kumar & Stephen T. Newman. *A Unified Manufacturing Resource Model for representing {CNC} machining systems*. Robotics and Computer-Integrated Manufacturing, vol. 25, no. 6, pages 999 – 1007, 2009. 18th International Conference on Flexible Automation and Intelligent Manufacturing.

[Vijayaraghavan 08] Athulan Vijayaraghavan, Will Sobel, Armando Fox, David Dornfeld & Paul Warndorf. *Improving machine tool interoperability using standardized interface protocols : MT connect*. Laboratory for Manufacturing and Sustainability, 2008.

[Vijayaraghavan 09] Athulan Vijayaraghavan & David Dornfeld. *Addressing process planning and verification issues with MTConnect*. Laboratory for Manufacturing and Sustainability, 2009.

[Vincent Wang 13] Xi Vincent Wang & Xun W Xu. *An interoperable solution for Cloud manufacturing*. Robotics and Computer-Integrated Manufacturing, vol. 29, no. 4, pages 232–247, 2013.

[Voas 09] Jeffrey Voas & Jia Zhang. *Cloud Computing : New Wine or Just a New Bottle ?* IT Professional, vol. 11, no. 2, pages 15–17, March 2009.

[Wang 02] Lihui Wang, Weiming Shen, Helen Xie, Joseph Neelamkavil & Ajit Pardasani. *Collaborative conceptual design – state of the art and future trends*. Computer-Aided Design, vol. 34, no. 13, pages 981–996, 2002.

[Wang 08] Lizhe Wang, Jie Tao, Marcel Kunze, Alvaro Canales Castellanos, David Kramer & Wolfgang Karl. *Scientific Cloud Computing : Early Definition and Experience.* In HPCC, volume 8, pages 825–830, 2008.

[Wang 12] X.V. Wang & X.W. Xu. *DIMP : an interoperable solution for software integration and product data exchange*. Enterprise Information Systems, vol. 6, no. 3, pages 291–314, 2012.

[Wang 14] Xi (Vincent) Wang. *Development of an Interoperable Cloud-based Manufacturing System*. PhD thesis, ResearchSpace@ Auckland, 2014.

[Wegner 96] Peter Wegner. *Interoperability*. ACM Computing Surveys (CSUR), vol. 28, no. 1, pages 285–287, 1996.

[Werner 06] Vogels Werner. *A Word on Scalability. Available online at : http ://www.allthingsdistributed.com/2006/03/a_word_on_scalability.html*, 2006.

[Williams 98] T Williams. *The Purdue enterprise reference architecture and methodology (PERA)*. Handbook of life cycle engineering : concepts, models, and technologies, vol. 289, 1998.

[Wood 09] Timothy Wood, Alexandre Gerber, KK Ramakrishnan, Prashant Shenoy & Jacobus Van der Merwe. *The case for enterprise-ready virtual private clouds*. In Proceedings of HotCloud '09 Workshop on Hot Topics in Cloud Computing, San Diego, CA, USA, 2009.

[Wooldridge 00] Michael Wooldridge, Nicholas R Jennings & David Kinny. *The Gaia methodology for agent-oriented analysis and design*. Autonomous Agents and multi-agent systems, vol. 3, no. 3, pages 285–312, 2000.

[Wu 10] L. Wu & C. Yang. *A solution of manufacturing resources sharing in cloud computing environment*. In Cooperative Design, Visualization, and Engineering, pages 247–252. Springer, 2010.

[Wu 12] Dazhong Wu, J Lane Thames, David W Rosen & Dirk Schaefer. *Towards a cloud-based design and manufacturing paradigm : looking backward, looking forward*. In ASME 2012 International Design Engineering Technical Conferences and Computers and Information

in Engineering Conference, pages 315–328. American Society of Mechanical Engineers, 2012.

[Wu 13a] Dazhong Wu, Matthew J Greer, David W Rosen & Dirk Schaefer. *Cloud manufacturing : drivers, current status, and future trends*. In ASME 2013 International Manufacturing Science and Engineering Conference collocated with the 41st North American Manufacturing Research Conference. American Society of Mechanical Engineers, 2013.

[Wu 13b] Dazhong Wu, Matthew John Greer, David W. Rosen & Dirk Schaefer. *Cloud manufacturing : Strategic vision and state-of-the-art*. Journal of Manufacturing Systems, vol. 32, no. 4, pages 564 – 579, 2013.

[Xu 08] Y. Xu, T. Xiao, C.E. Liang & L. Zhang. *Federated integration of networked manufacturing service platforms*. Advanced Engineering Informatics, vol. 22, no. 3, pages 317–327, July 2008.

[Xu 09] Xun Xu & Andrew Yeh Ching Nee. Advanced design and manufacturing based on step. Springer Science & Business Media, 2009.

[Xu 12a] W. Xu, Z. Zhou, D.T. Pham, Q. Liu, C. Ji & W. Meng. *Quality of service in manufacturing networks : a service framework and its implementation*. The International Journal of Advanced Manufacturing Technology, vol. 63, no. 9-12, pages 1227–1237, 2012.

[Xu 12b] X. Xu. *From cloud computing to cloud manufacturing*. Robotics and Computer-Integrated Manufacturing, vol. 28, no. 1, pages 75–86, February 2012.

[Xu 12c] Xun Xu. *From cloud computing to cloud manufacturing*. Robotics and Computer-Integrated Manufacturing, vol. 28, no. 1, pages 75–86, February 2012.

[Yang 03] H Yang & D Xue. *Recent research on developing Web-based manufacturing systems : a review*. International Journal of Production Research, vol. 41, no. 15, pages 3601–3629, 2003.

[Yang 09] Dong Yang, Rui Miao, Hongwei Wu & Yiting Zhou. *Product configuration knowledge modeling using ontology web language*. Expert Systems with Applications, vol. 36, no. 3, pages 4399–4411, April 2009.

[Zellner 11] Gregor Zellner. *A structured evaluation of business process improvement approaches*. Business Process Management Journal, no. 2, pages 203–237, April 2011.

[Zeng 02] L. Zeng, D. Flaxer, H. Chang & J.J. Jeng. *PLM flowâDynamic Business Process Composition and Execution by Rule Inference*. In Technologies for E-Services, pages 141–150. Springer, 2002.

[Zhang 00] Y.P Zhang, C.C Zhang & H.P. Wang. *An Internet based STEP data exchange framework for virtual enterprises*. Computers in Industry, vol. 41, no. 1, pages 51–63, 2000.

[Zhang 10a] Lin Zhang, H Guo, F Tao, YL Luo & N Si. *Flexible management of resource service composition in cloud manufacturing*. In Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on, pages 2278–2282. IEEE, 2010.

[Zhang 10b] Lin Zhang, Yong-Liang Luo, Fei Tao, Lei Ren & Hua Guo. *Key technologies for the construction of manufacturing cloud*. Computer Integrated Manufacturing Systems, vol. 16, no. 11, pages 2510–2520, 2010.

[Zhang 14a] Lin Zhang, Yongliang Luo, Fei Tao, Bo Hu Li, Lei Ren, Xuesong Zhang, Hua Guo, Ying Cheng, Anrui Hu & Yongkui Liu. *Cloud manufacturing : a new manufacturing paradigm*. Enterprise Information Systems, vol. 8, no. 2, pages 167–187, March 2014.

[Zhang 14b] Lin Zhang, Yongliang Luo, Fei Tao, Bo Hu Li, Lei Ren, Xuesong Zhang, Hua Guo, Ying Cheng, Anrui Hu & Yongkui Liu. *Cloud manufacturing : a new manufacturing paradigm*. Enterprise Information Systems, vol. 8, no. 2, pages 167–187, 2014.

[Zhao 99] J. Zhao, W.M. Cheung & R.I.M. Young. *A consistent manufacturing data model to support virtual enterprises*. International Journal of Agile Management Systems, vol. 1, no. 3, pages 150–158, 1999.

[Zimmermann 08] Antoine Zimmermann. *Sémantique des réseaux de connaissances : gestion de l'hétérogénéité fondée sur le principe de médiation*. Theses, Université Joseph-Fourier - Grenoble I, November 2008.

Titre Proposition d'une modélisation unifiée du Cloud Manufacturing et d'une méthodologie d'implémentation, basées sur les ontologies d'inférence.

Résumé Dans ces travaux de recherche, nous proposons une méthodologie pour l'élaboration d'une architecture Cloud Manufacturing. Le CM est un paradigme émergeant dans lequel des ressources dynamiques virtualisées et extensibles sont fournies à un utilisateur comme services à travers internet. Notre architecture est utilisée comme une plateforme pour mettre en correspondance des utilisateurs et des fournisseurs de ressources manufacturières pouvant collaborer ensemble tout au long du cycle de vie du produit tout en réduisant les coûts et le temps de développement. Comme certains vendeurs peuvent décrire leurs services de différentes façons, les technologies du web telles que les ontologies sont des outils robustes pour mettre en correspondance les descriptions des vendeurs avec les requêtes des utilisateurs. Nous utilisons les ontologies pour élaborer le modèle de connaissance du domaine du Cloud Manufacturing. L'ontologie définit les étapes du cycle de vie du produit comme des services et prend en compte les caractéristiques du Cloud Computing (stockage, capacité de calcul, etc.). L'ontologie CM contribue à automatiser la découverte des services et est incluse dans la plateforme pour mettre en correspondance des utilisateurs et les fournisseurs. La méthodologie proposée (ASCI-Onto) est inspirée par le cadre méthodologique ASCI qui a déjà été utilisé pour l'étude de systèmes logistiques, hospitaliers et de production. L'objectif de cette nouvelle méthodologie est de développer facilement une librairie de composants pour un système CM. Un exemple d'application avec un modèle de simulation, basé sur le logiciel CloudSim, est présenté. L'objectif est d'aider les industriels à prendre leurs décisions pour concevoir un système CM.

Mots-clés Cloud Manufacturing, PLM, Méthodologie ASCI, Ontologie.

Title A unified Cloud Manufacturing model and methodology based on ontologies.

Abstract In this research, we introduce a methodology to build a CM architecture. Cloud Manufacturing is an emerging paradigm in which dynamically scalable and virtualized resources are provided to the users as services over the Internet. Our architecture serves as a platform for mapping users and manufacturing resources' providers with the aim of enhancing collaboration within Product Lifecycle Management (PLM) by reducing costs and development time. Since some providers may use different descriptions of their services we believe that semantic web technologies like ontologies are robust tools for mapping providers' descriptions and users' requests in order to find the suited service. Then, we use ontology to build the knowledge model of Cloud Manufacturing domain. The ontology defines the steps of the product lifecycle as services and also takes in account the Cloud Computing features (storage, Computing capacity, etc.). The Cloud Manufacturing ontology contributes to intelligent and automated service discovery and is included in a platform for mapping users and providers. The proposed methodology ASCI-Onto is inspired by ASDI framework (analysis-specification-design-implementation) which has already been used in supply chain, healthcare and manufacturing domains. The goal of the new methodology consists of designing easily a library of components for a Cloud Manufacturing system. Finally, an example of application of this methodology with a simulation model, based on the CloudSim software, is presented. The goal is to help industrials to make their decisions in order to design Cloud Manufacturing systems.

Keywords Cloud Manufacturing, PLM, ASDI methodology, Ontology

ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY